



Anuraj Rajendraprakash

Terrain mapping near the vehicle, SLAM and global map building for lunar rover

School of Electrical Engineering

Department of Automation and Systems Technology

Thesis submitted in partial fulfilment of the requirements for the degree of
Master of Science in Technology

Espoo, August 20, 2013

Instructor:

Dr. Sami Terho
Aalto University
School of Electrical Engineering

Supervisors:

Professor Emeritus Aarne Halme
Aalto University
School of Electrical Engineering

Professor Thomas Gustafsson
Luleå University of Technology
Department of Computer Science,
Electrical and Space Engineering

Acknowledgements

First of all I would like to thank Professor Emeritus Aarne Halme, for supervising my thesis and providing feedback for it. I would also like to thank Professor Thomas Gustafsson for his feedback.

This thesis was done as a part of the project aiming to compete in the annual NASA *Lunabotics* competition. It was on Tomi Ylikorpi's(Lic.Sc.(Tech.)) suggestion that I had undertaken work on this project as a part of my thesis. Tomi has been helping us since the time we were admitted to Aalto University for various academic and non-academic activities. Tomi also designed the mechanical four-bar link used in the thesis. I am really grateful to Tomi for all the help and guidance during my time here at Aalto University.

I sincerely thank my instructor, Dr. Sami Terho for his guidance, support and regular feedback on the thesis work. Sami's regular feedback was of immense help in creating this manuscript. He not only helped me in design and set-up of the experiments and but also in collecting data from the experiments. I am also grateful to Laura Mendoza for her feedback which helped me create the flow of ideas in this thesis. I wish to express my gratitude to my friend Jaime Hernandez Zarate for all the technical discussions that helped me finish the thesis on time. I would also like to thank the team members of the Aalto Lunabotics team for the great times we had while working in the Lunabotics project. I thank my friends Suneet Gupta and Aditya Saxena for taking time out of their busy schedule to proof read my thesis.

Dr. Annika Salama has been helping us with the administrative work of the university since we have arrived here in Aalto. I thank her for all the help. I

am grateful to Matthieu Myrsky for his help with the Pioneer mobile platform used in the thesis. I thank Sami Kielosto for his help with procuring electronics for the thesis. I am grateful to Matti Lassila from The Finnish School of Watchmaking(Micromechanics programme) for manufacturing the four-bar link mechanism used in the thesis work. I also thank all the staff of the Automation Technology Laboratory for making the time spent in the laboratory enjoyable.

I wish to thank the European Union Education, Audiovisual and Culture Executive Agency that provides the funding for the Erasmus Mundus Masters Programmes. This masters programme has enabled to make friends from six different continents. I sincerely thank them for the scholarship provided to me for my studies that enabled me to revel in the joys of learning and research. I am also grateful to be a part of the SpaceMaster Round 7 family with whom I had one of the best times of my life in Kiruna, Sweden.

Isaac Newton had once said, “If I have seen further, it is by standing on the shoulders of Giants”. This thesis work has been possible because of the work of many researchers around the world who write open source softwares. I am thankful to the people around the world working to stretch the limits of knowledge in robotics and whose work has helped me understand and carry out this thesis.

I have been extremely lucky in having met and be friends with amazing people from SVNIT, Surat, India during my bachelor studies and during my SpaceMaster studies. I would always be grateful for it.

Lastly I would like to thank my mother, my father and my brother for their unconditional love and support without which the SpaceMaster journey would not be possible.

Espoo, August 20, 2013

Anuraj Rajendraprakash

Author:	Anuraj Rajendraprakash		
Title of the thesis:	Terrain mapping near the vehicle, SLAM and global map building for lunar rover		
Date:	August 20, 2013	Number of pages:	67
Department:	Automation and Systems Technology		
Programme:	Master's Degree Programme in Space Science and Technology		
Professorship:	Automation Technology (AUT-84)		
Supervisors:	Professor Emeritus Aarne Halme (Aalto) Professor Thomas Gustafsson (LTU)		
Instructor:	Dr. Sami Terho		
<p>There has been increasing interest to go back to the moon in the recent past because of various scientific and socio-economic reasons. In order to go back to the moon there is a need to study the lunar environment. Although having a permanent mission outpost on the moon is the final goal it is better to send mobile rovers to the surface of the moon first to study lunar environment before starting the human missions to moon again. With the increasing autonomous mobility of the lunar rovers some aspects become increasingly important namely localization, navigation and mapping. Although the two-dimensional localization and mapping algorithms are becoming more and more mature for indoor mobile robotics, they cannot be used, as is, for autonomous lunar rovers. The terrain on the Moon is not even and would have various kinds of obstacles for the rovers to manoeuvre and traverse. Moreover, environmental features like walls and corners are not available in the environment in which the rovers would have to navigate. In such environments it becomes important for the rover to have the ability to map its surrounding in three dimensions. Although LIDAR based systems have not been widely used on actual lunar missions for mapping yet, they have the advantage of being more accurate and long-range. The focus of this thesis would be to develop and equip a lunar rover prototype with the three-dimensional terrain mapping ability using Light Detection and Ranging (LIDAR) sensor which would help the rover to traverse its environment without collisions. A three-dimensional point cloud was used to map the environment using the Iterative Closest Point(ICP) algorithm.</p>			
Keywords: Lunar-rover, Terrain mapping, SLAM, LIDAR, Point-cloud registration, Iterative Closest Point(ICP)			

Contents

1	Introduction	1
1.1	Going back to the Moon	1
1.2	Lunar environment	3
1.3	Mapping and navigation for lunar rovers	4
1.4	Motivation and objectives	4
1.5	Outline	6
2	Related Work	7
2.1	Sensing systems for mapping and navigation	7
2.1.1	Dead-reckoning	7
2.1.2	Cameras	8
2.1.3	2D LIDAR scanner	9
2.1.4	3D LIDAR scanner	10
2.1.5	Star-trackers	10
2.2	Robotic mapping	11
2.2.1	Classification of robotic maps	11
2.2.1.1	Metric maps	12
2.2.1.2	Topological maps	13
2.2.1.3	2D vs 3D maps	13
2.2.2	Localization and mapping	14
2.2.3	Robotic mapping approaches	15
2.2.3.1	Kalman Filter approaches	15

2.2.3.2	Expectation Maximization approaches	15
2.2.3.3	Particle Filter approaches	16
2.2.3.4	Scan Matching approaches	16
2.3	Mapping in planetary rovers	17
3	Implementation	20
3.1	Hardware platforms and software	20
3.1.1	Hardware platforms	20
3.1.1.1	Mobile platform	20
3.1.1.2	LIDAR scanner	21
3.1.1.3	Micro-controller and servo	21
3.1.2	Software	21
3.1.2.1	Robot Operating System	21
3.1.2.2	Point Cloud Library	23
3.2	System overview	24
3.2.1	ROS stacks/packages used in the system	25
3.3	Tilting system	28
3.3.1	Servo control with ROS and Arduino	28
3.3.2	Four-bar link mechanism	29
3.4	Assembling laser-scans to point-clouds	31
4	Global map building with ICP-based SLAM	36
4.1	The Iterative Closest Point algorithm	36
4.1.1	Finding closest point using kD-tree	38
4.1.2	Error minimization	39
4.2	Map representations	40
4.3	Data filtering	41
5	Experiments and Results	46
5.1	Experimental set-up	46

5.2	Registration without an initial guess	46
5.3	Registration with initial guesses	48
5.3.1	Scenario 1: Indoor room mapping	48
5.3.2	Scenario 2: Indoor corridor mapping	51
5.3.3	Scenario 3: Indoor unstructured room mapping	51
5.3.4	Scenario 4: Outdoor mapping	51
5.3.5	Octree compression results	53
6	Conclusions and Future Work	57
6.1	Conclusions	57
6.2	Future Work	58
	References	61
A	Four-bar linkage position analysis	I
B	Flow diagrams	V

List of Tables

4.1	Advantages and disadvantages of different map representations (Hornung et al., 2013)	42
5.1	Compression results of maps generated by <code>Octomap</code> framework .	56

List of Figures

1.1	Lunar and Martian rovers	2
1.2	Image showing surface of the Moon taken during Apollo mission. Image courtesy National Aeronautics and Space Administration, USA (NASA). Scanning credit JSC and Kipp Teague	5
2.1	A simple stereo geometry. Figure courtesy Kyrki (2012)	8
2.2	Working of Two Dimensional (2D) LIDAR scanner	9
2.3	2D LIDAR scanner articulation along different axes to obtain Three Dimensional (3D) point-cloud. Figure from Wulf and Wag- ner (2003)	10
2.4	Principle of Star-tracker. Figure adapted from NASA (2013a) . .	11
2.5	Difference between metric and topological maps. Figure from Fil- liat and Meyer (2003)	11
2.6	Feature map representation. Figure from Filliat and Meyer (2003)	12
2.7	Occupancy grid showing free space map representation. Figure from Filliat and Meyer (2003)	13
3.1	Pioneer 3DX. Image from Pioneer (2013)	20
3.2	Pitching scan and the resulting cloud. Figure from Wulf and Wag- ner (2003)	21
3.3	Servo and micro-controller	22
3.4	Broad overview of the software nodes.	25
3.5	Node graph of the system	26
3.6	Working of the <code>laser_scan_assembler</code> node. Figure from ROS- Wiki-laser-assembler (2013)	28

3.7	Arduino connection with servo. Figure from ROS-Wiki-Arduino (2013)	29
3.8	Four-bar crank-rocker mechanism in two different positions . . .	30
3.9	Generalized crank-rocker four-bar linkage	31
3.10	The mechanism showing horizontal position of LIDAR scan-plane	32
3.11	The mechanism showing tilted position of LIDAR scan-plane . .	32
3.12	Photos of the actual mechanism built	33
3.13	The URDF model visualized with laser in horizontal position . .	34
3.14	The URDF model visualized with laser in tilted position	35
4.1	kD-tree of the order 2. Figure adapted from Sedgewick and Wayne (2013)	39
4.2	Classic error metrics for ICP. Figures from Park and Subbarao (2003)	40
4.3	Different types of map representation. Figures from Hornung et al. (2013)	41
4.4	Radius Outlier Filtering. Figure adapted from PCL-Documentation (2013a)	45
5.1	Registered aligned point-cloud of half of the room without odometry (Color legend unit in meters)	47
5.2	Registered unaligned point-cloud of half of the room without odometry (Color legend unit in meters)	47
5.3	Photograph of half of the experimental room	48
5.4	Registered point-cloud of the whole room. Top View (Color legend unit in meters)	49
5.5	3D point-cloud of the view relating to Figure 5.6 (Color legend unit in meters)	49
5.6	Photograph of room from one of the Views	50
5.7	Top view of the map obtained by moving platform. (Color legend unit in meters)	50

5.8	3D map obtained by moving the platform. View related to Figure 5.6 (Color legend unit in meters)	51
5.9	Photograph of corridor scenario	52
5.10	3D map of the corridor (Color legend unit in meters)	52
5.11	3D octree map of the Corridor	53
5.12	Photograph of unstructured room scenario	53
5.13	3D Map of the unstructured room (Color legend unit in meters)	54
5.14	3D Octree Map of unstructured room	54
5.15	Photograph of outdoor scenario	55
5.16	3D Map of the outdoor scene	55
5.17	3D Octree Map of outdoor scene	56
A.1	A general four-bar mechanism	I
B.1	Flow Chart of the Arduino Node	V
B.2	Flow Chart of the Angle Publisher Node	VI
B.3	Node Graph of the System	VII

List of Algorithms

- 1 The Iterative Closest Point (ICP) algorithm from Nüchter (2009) 37

Listings

3.1 URDF for the platform used	34
--	----

Symbols and Abbreviations

\hat{D}	A three-dimensional point-set name D
N_d	The number of in points in the point-set D
\mathbf{R}	Rotation Matrix
\mathbf{t}	Translation Vector
m_i	The i^{th} point from the point-set \hat{M}
\mathbf{m}_i	Position Vector of the the point m_i
$\ \hat{\mathbf{m}}_i\ $	Norm of the Vector \mathbf{m}_i
$sgn(w)$	Signum Function of the variable w

2D Two Dimensional

3D Three Dimensional

APS Active Pixel Sensors

ARIA Advanced Robot Interface for Applications

BSD Berkeley Software Distribution

CSA Canadian Space Agency

EM Expectation Maximization

GESTALT Grid-based Estimation of Surface Traversability Applied to Local Terrain

ICP Iterative Closest Point

JSC Johnson Space Center, NASA

LIDAR Light Detection and Ranging

MER Mars Exploration Rovers

MET Mars Emulation Terrain

NASA National Aeronautics and Space Administration, USA

PCL Point Cloud Library

PWM Pulse Width Modulation

ROS Robot Operating System

SLAM Simultaneous Localization and Mapping

UART Universal Asynchronous Receiver/Transmitter

URDF Unified Robot Description Format

USB Universal Serial Bus

WFOV Wide Field Of View

Chapter 1

Introduction

“Every surviving civilization is obliged to become space-faring – not because of exploratory or romantic zeal, but for the most practical reason imaginable: staying alive...”

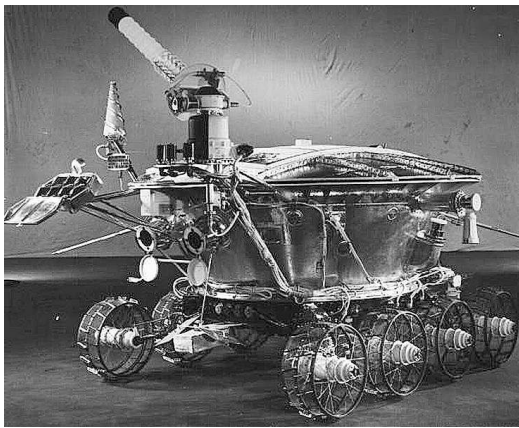
- Carl Sagan

1.1 Going back to the Moon

Humans have been sending space-crafts to various parts of the solar system such as the Moon and other planets since the 1960s. The Voyager-1 and Voyager-2 missions launched in 1977 have reached the Heliosheath, which is the outermost layer of the solar system (NASA, 2013b). In spite of having sent missions to such far-off distances, humans are far from setting up outposts in the solar system and thus becoming a space-faring civilization. A promising candidate to set up human outposts in the solar system is the Moon because of its proximity to the Earth. However, the Moon has a very harsh environment, which would provide many difficulties for setting up an outpost on it. It is necessary to collect as much information about the Moon as possible, in order to have safe manned missions to the Moon. The scientific information collected would also help in increasing our understanding about the solar system. A good way to collect scientific information about the Moon is to first send autonomous robotic missions. In addition to providing valuable information about the Moon, these robotic missions could also do the preliminary work of setting

up the outpost.

The concept of mobile robotic missions was established by the Lunakhod-1, which was the first successful lunar rover traversing on the surface of the moon (Huntress et al., 2003, p. 566; Blair, 2011, p.50). Exploratory rovers have not just been sent to the moon but also to our neighbouring planet Mars. NASA was successful in landing *Sojourner* rover on the surface of Mars in July, 1997 making it the first planetary rover to traverse successfully on another planet (Bajracharya et al., 2008). The latest mission to land a rover on Mars was the Mars Science Laboratory (MSL) which landed the *Curiosity* rover in August, 2012 near Gale Crater on Mars. After the initial spike of interest in lunar missions during the 1960s, there has been a renewed interest in the Moon in the last decade. In 2006, NASA conducted a survey about the reasons to go back to the Moon. From this survey NASA concluded that there are six major themes which are of common interest among the world's nations and which justify the reasons to go back to the Moon. They are a) Human Civilization; b) Scientific Knowledge; c) Exploration Preparation; d) Global Partnerships; e) Economic Expansion f) Public Engagement.



(a) Lunokhod-1



(b) Mock-up of MER, Sojourner and Curiosity

Figure 1.1: Lunar and Martian rovers

Crawford et al. (2012) mention that none of the various missions to the Moon during the last decade landed a spacecraft on the surface of the moon in a guided manner. It is best to send robotic missions to the moon first to collect scientific information before sending manned missions again. A mission to send

autonomous planetary rovers to the moon would require solving several technical challenges because of the extreme environmental conditions on the moon.

1.2 Lunar environment

The lunar environment is extreme for robotic missions due to several factors. Vasavada et al. (2012) observe that the lack of atmosphere, the presence of insulating regolith layer and the slow rotation of the Moon make the temperature on the surface of the Moon vary from 400 Kelvin during daytime to 100 Kelvin at night. Grün et al. (2011) mention that the lunar surface is covered with several meters of regolith which could be up to 10 meters and consist of rocks, pebbles and dust. The lunar dust provides another set of problems for the lunar rover and missions. The lunar dust is “electro-statically charged, is difficult to remove and appears to get everywhere” (DiGiuseppe et al., 2009). The rotation of the Moon around the Earth also makes some parts of the Moon permanently remain in the dark and other parts in bright sun light. In addition to these differences, the Moon does not have the kind of magnetic field as the Earth does.

This extreme environment of the Moon rules out the use of many types of sensors that can be easily used on the surface of the Earth. The terrain on the Moon is not even and the regolith makes the terrain smooth and slippery. As a result, wheel odometry is not very useful. Sonar cannot be used because of no atmosphere on the surface of the Moon. Magnetometers do not work well for navigation because of the lack of magnetic field similar to Earth. Nishida and Wakabayashi (2010) argue that using the sun sensor is difficult because the sun may not be visible in all parts of the Moon at all times. They add that a lunar rover could use a combination of the following sensors for mapping and navigation: a) Fibre optic gyros b) Star Tracker c) Stereo Cameras d) Scanning Laser Range Finder

1.3 Mapping and navigation for lunar rovers

With the increasing need for lunar rover mobility, some aspects of robotics become increasingly important namely mapping, localization and navigation. It is absolutely essential to have good mapping, localization and navigation if the robot has to move around without collisions. Such collisions could prove to be fatal to the mission if the robot or rover is being sent for space missions to the Moon.

The mapping and localization algorithms used for indoor mobile robotics are becoming increasingly mature but they cannot be directly used for mapping large-scale planetary environments. Typically two-dimensional mapping is sufficient for indoor mobile robotics, but in lunar environments features like walls are not present to help in mapping and localization. The Figure 1.2 on the following page shows the image of the surface of the Moon taken during the Apollo missions. The mapping and navigation algorithms used in outdoor environment are comparatively more suitable for lunar rovers but they would have to be modified to suit the needs of the lunar rover.

1.4 Motivation and objectives

A three-dimensional map of the surrounding environment would be more useful for navigation for a lunar rover. Making a reasonably accurate model of its surrounding environment is one of the important functions that are needed for autonomous navigation (Lacroix et al., 2002). The two sensors that are good candidates for environment modelling by planetary rovers are: a) Stereo Vision Systems b) Systems with LIDAR using Lasers. Of the above two sensing systems, passive stereo vision sensors were used on Mars Exploration Rovers (Maimone et al., 2006). Passive stereo vision systems have small field of views and their dependence on external light sources hampers their performance under varying lighting conditions making their use difficult under changing illumination (Barfoot et al., 2011). LIDAR based sensing systems has been used by Ishigami et al. (2012) on a prototype lunar rover. Stereo vision systems have the advantage of being light weight, requiring low power and very few moving



Figure 1.2: Image showing surface of the Moon taken during Apollo mission. Image courtesy NASA. Scanning credit JSC and Kipp Teague

parts (Maimone et al., 2006) and hence they have been frequently used in space missions. On the other hand, LIDAR systems have the advantage of being more accurate and long-range. As LIDAR based systems have more moving parts, it is difficult and expensive to make space-qualified LIDAR sensors. With the advent of technology, LIDAR based space-qualified systems for three-dimensional mapping of lunar and planetary environment would become available (Barfoot et al., 2011). There is an increasing need to do more research on developing space-qualified sensing systems. At the same time, it is also essential that more research be done on the various algorithmic techniques in order to use data from three-dimensional LIDAR sensors. The more efficient algorithms would be useful in the constrained resources available for computation on the lunar rover. This thesis aims to investigate one of algorithms for suitability on the lunar rover. This broad aim can be broken down into the following objectives:

1. To study the different types of mapping approaches and resulting maps.
To select a suitable mapping algorithm based on the study.
2. To develop a system that would enable a mobile platform to generate 3D terrain scans.

3. To combine the terrain scans into a globally consistent 3D map using the selected algorithm.
4. To test the 3D mapping system developed, in lunar like terrains.

1.5 Outline

Chapter 2 starts with a brief overview of the sensors that could be used on the lunar rover. This is followed by the explanation of various kinds of maps. Then the various methods to make robotic maps are explained. Finally there is a discussion about the robotic planetary mapping.

Chapter 3 starts with the brief description of the various hardware and softwares used for the thesis. Then a description of the tilting system is given. This tilting system was used to get a 3D point cloud with a 2D LIDAR scanner. The chapter ends with the explanation of the software part of the tilting system controlling the mechanism to get the terrain scans.

The Iterative Closest Point (ICP) algorithm that was used for global map building in the thesis work is explained in Chapter 4. The algorithm explanation is followed by a brief description of the various map representations that are used in robotics along with their advantages and disadvantages. The chapter ends with the different types of data filters that were used for outlier rejection and removing redundant points.

The experiments that were carried out to test the algorithm and their resulting maps are presented in the Chapter 5.

Chapter 6 summarises the work of the thesis with conclusions and ends with the future work possible to improve the work done in the thesis.

Chapter 2

Related Work

One of the primary requirements for good autonomous navigation for planetary rovers or mobile robots is that it should have a reliable and robust sensing system. A good sensing system is crucial for three essential functions that a planetary rover or mobile robot has to perform: a) pose estimation b) pose maintenance c) map construction (Dudek and Jenkin, 2010, p. 82). The following section discusses some sensing systems that could be used on lunar rover.

2.1 Sensing systems for mapping and navigation

2.1.1 Dead-reckoning

According to Everett (1995, p.35), dead-reckoning is a mathematical method to deduce the present location of a mobile robot by incrementing the known position and velocity over a specific period of time, through a known course. Fuke and Krotkov (1996) define dead-reckoning as "a navigation method based on measurements of distance travelled from a known point used to incrementally update the robot pose". They say that the primary advantage of dead-reckoning is that it is simple and cheap to build and use, but, has the disadvantage of having an unbounded accumulation of errors.

Inertial navigation is one of the methods that could be used for dead-reckoning. The idea behind inertial navigation is to measure small accelerations in all the three principal axes and to integrate the accelerations to obtain the velocity and position information. Although, the concept of inertial navigation seems to be fairly simple, minimizing the various sources of error makes the practical implementation difficult (Everett, 1995, pp.47-48). Wheel odometry is another method for dead-reckoning. The pose of the rover or mobile robot could be deduced by counting the number of rotations of the wheels. In wheel odometry the errors may creep in because of slippage. These errors in dead-reckoning methods become unbounded in a short period of time, if they are not corrected periodically. Hence dead-reckoning methods are not used alone but always in addition to some other sensors to correct the errors.

2.1.2 Cameras

The ability of vision for a robot with a camera is a powerful sensing medium. However, vision is really difficult to use from the perspective of robotics relatively (Dudek and Jenkin, 2010, p. 123). Stereo cameras can be used to obtain the depth information of surroundings. The images from the two cameras can be analysed to get disparity information. Disparity is defined as the “difference in retinal position between the corresponding points in the two images” (Trucco and Verri, 1998). It can be easily proved that depth is inversely proportional to disparity. Thus using disparity the depth information can be found out.

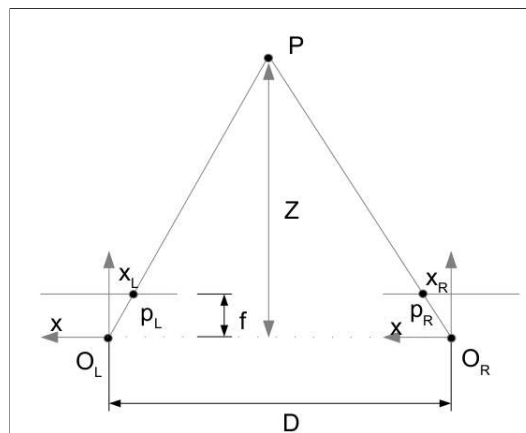


Figure 2.1: A simple stereo geometry. Figure courtesy Kyrki (2012)

The Figure 2.1 on the previous page shows a simple stereo setup. O_r and O_l are the optical centres of the two cameras. f is the focal length. Z is the depth of the point P being observed. p_r and p_l are the images of the point P on the right and left image planes respectively. D is the distance between the optical centres. x_r and x_l are the distances of the image points from the optical axes of the two camera systems respectively. It can be shown that:

$$Z = f \frac{D}{d} \quad (2.1)$$

where $d = x_r - x_l$. Here d is known as the disparity.

2.1.3 2D LIDAR scanner

A 2D LIDAR scanner is made by sweeping a laser beam in a plane with the help of a mirror. The Figure 2.2 shows the working of a 2D laser scanner. The figure has three parts: the top part shows how a laser beam is deflected using a rotating mirror to obtain a 2D scan; the middle part shows the top view of the laser set-up in its environment; the bottom part shows the 2D scan obtained.

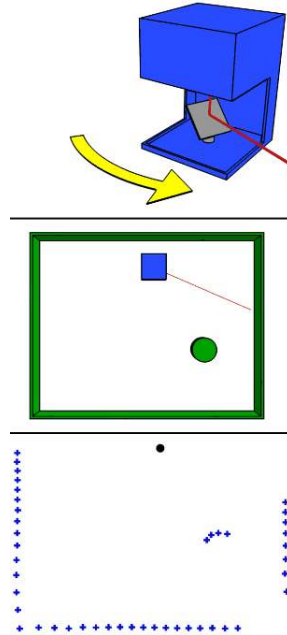


Figure 2.2: Working of 2D LIDAR scanner

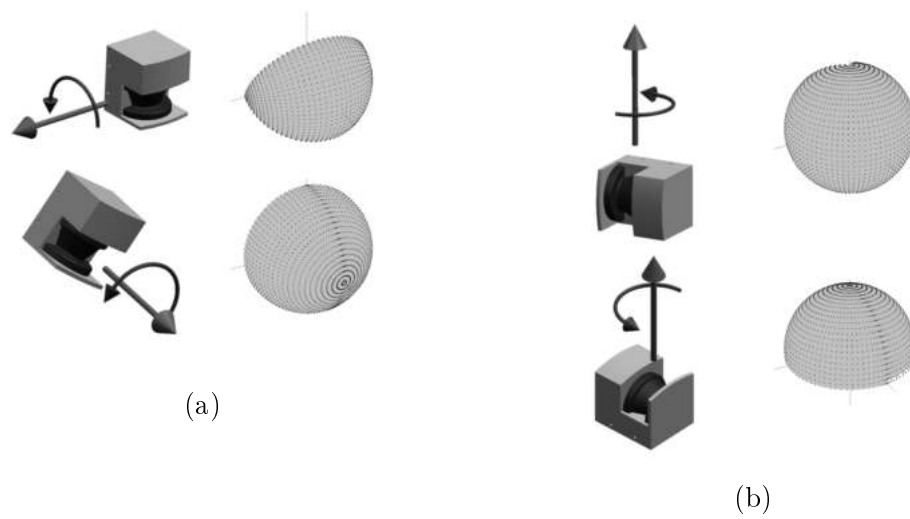


Figure 2.3: 2D LIDAR scanner articulation along different axes to obtain 3D point-cloud. Figure from Wulf and Wagner (2003)

2.1.4 3D LIDAR scanner

The concept of 3D LIDAR is similar to that of a 2D LIDAR except that the former has additional information about the third dimension. The information from a 3D LIDAR scanner is in the form of a point of clouds. Highly accurate 3D LIDAR systems are very expensive and this impacts its large-scale use (Nüchter, 2009, pp. 12-14). A 3D scanner can be built from a less expensive 2D LIDAR scanner by articulating the 2D LIDAR scanner along an axis. A few ways of obtaining the a 3D laser scan by articulating a 2D LIDAR scanner is shown in the Figure 2.3 and the resulting 3D point-cloud is shown beside each of them.

2.1.5 Star-trackers

Star-tracker are basically cameras with wide field-of-view. Star-trackers use cameras with Active Pixel Sensors (APS) and Wide Field Of View (WFOV) (NASA, 2013a). The camera takes pictures of the star patterns and then these pictures are compared with an internal database. This comparison helps in knowing approximate location of the robot or spacecraft (NASA, 2013a). These steps are shown in the Figure 2.4 on the following page.

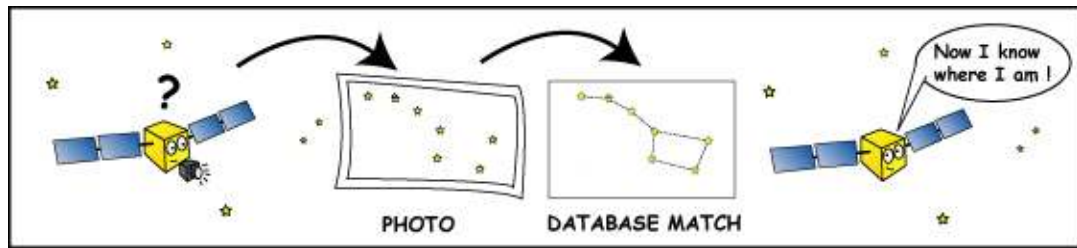


Figure 2.4: Principle of Star-tracker. Figure adapted from NASA (2013a)

2.2 Robotic mapping

2.2.1 Classification of robotic maps

The two main categories of robotic maps are: a) Metric Maps and b) Topological Maps. The geometric properties of the environment are used to build a metric map while the information about connectivity between different positions is used to build a topological map (Thrun, 2002).

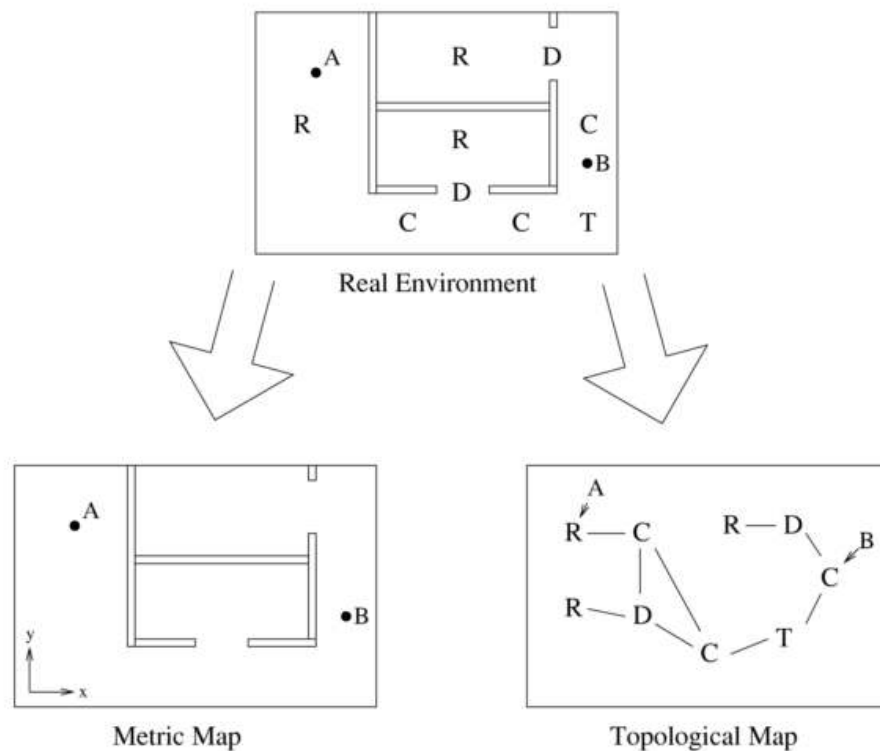


Figure 2.5: Difference between metric and topological maps. Figure from Filliat and Meyer (2003)

2.2.1.1 Metric maps

In metric maps, according to Filliat and Meyer (2003), the environment is modelled with a set of objects recognised and their coordinates in a Euclidean space. They say that metric maps are independent from the fact that it was made by a particular robot and a map made by one robot can be used by another robot. They also mention that metric maps are relatively easier to build than topological maps because of explicit information about the location of various objects. They add that one of the key differences from the topological maps is the use of sensor models which helps in fusion of data from proprioceptive and exteroceptive sensors.

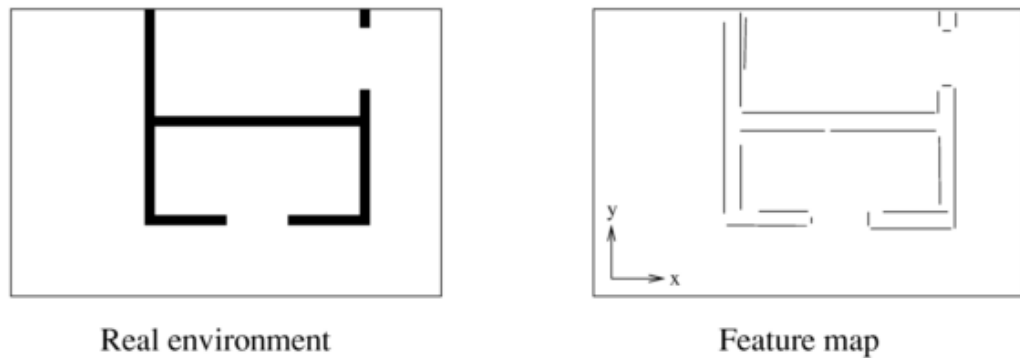


Figure 2.6: Feature map representation. Figure from Filliat and Meyer (2003)

Metric maps can be further classified into: a) Feature representation and b) Free space representation (Filliat and Meyer, 2003). In feature representation, various kind are features are extracted from the sensory data and their location is added to the map. However, in free-space representation, the environment model consists of tiny cells with the each cell having the probability of being empty or filled. The occupancy grid is an example of free space representation. The Figure 2.6 shows the example of a feature map representation while Figure 2.7 on the following page shows and occupancy grid map which is free-space map representation.

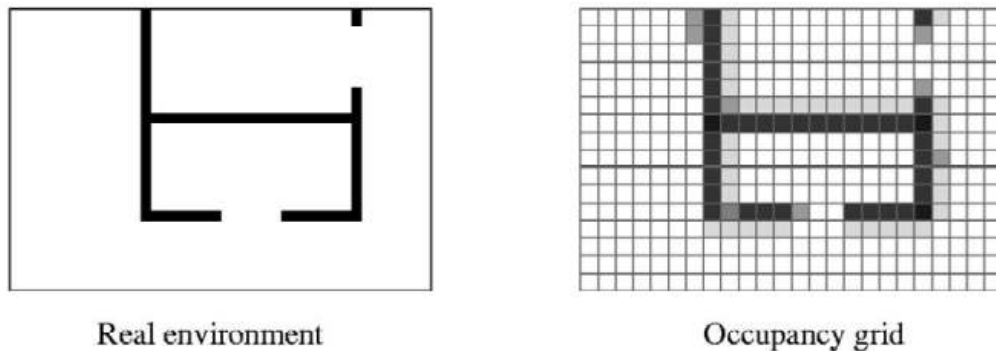


Figure 2.7: Occupancy grid showing free space map representation. Figure from Filliat and Meyer (2003)

2.2.1.2 Topological maps

In topological map representation, the environment is modelled as a collection of unique places and information about how the robot can travel between these places (Thrun, 2002; Filliat and Meyer, 2003, pp. 249-250). An advantage of using topological maps, as mentioned by Filliat and Meyer (2003), is that only certain places of the environment are learnt and this makes the topological maps sparse. They mention the other two advantages of topological maps as: a) Easier path planning process as the search space is small. b) No sensor model is required for building a topological map. However, they add that these advantages are outweighed in mapping large environments, which would require a more exhaustive exploration for higher accuracy of the position estimation. Additionally, they say that the sensor noise has a huge effect on the map made because incorrect sensor reading could lead to wrong identification of places and in large environments these errors in the maps would be difficult to correct.

2.2.1.3 2D vs 3D maps

At present the 2D mapping techniques are sufficiently mature for many indoor mobile robotics applications. Many of the SLAM methods in 2D are fast enough to be used in real time and to handle dynamic obstacles. However, in outdoor environments where there are no definite structures like walls, 2D mapping is not sufficient. For desert like environment similar to that on the Moon, a 3D map is much more useful. However, at present there are other challenges

for using 3D mapping systems. Firstly, 3D mapping sensors are much more expensive than 2D mapping sensors. Secondly the amount of data produced by 3D sensors requires a lot of processing power and could be a bottleneck for their use on a planetary rover with limited computational power. With the progress of technology, the price of the 3D sensor would go down and the computational power of the space-qualified processors would increase. This should enable to overcome the bottleneck discussed above.

2.2.2 Localization and mapping

In order to navigate in its environment the planetary rover requires a map of the environment. Mapping involves making a spatial model of the environment with the help of the sensors (Thrun, 2002). According to Meyer and Filliat (2003), navigation involves primarily three processes:

- *Map learning*, in which the data from the various sensors is stored for further use
- *Localization*, in which the present position of the rover or mobile robot is determined with the help of current map
- *Path Planning*, in which a suitable set of actions is chosen to reach the target goal from the current position

In classical robotic mapping which were popular before the late 1980s, the Map Learning and Localization steps were considered to be separate steps. But since the early 1990s the approaches which estimate the map and the pose of the robot simultaneously using probabilistic techniques have been extensively researched because they produce comparatively more accurate maps. This challenge of trying to achieve the localization and map building at the same time is called *Simultaneous Localization and Mapping (SLAM)* in literature. In SLAM approaches localization and map-learning are coupled processes because localization needs a map to know the current position while map building entails position estimation in the known map (Meyer and Filliat, 2003).

The next section briefly explains the various SLAM approaches used for making robotic maps.

2.2.3 Robotic mapping approaches

2.2.3.1 Kalman Filter approaches

In Kalman Filter approaches, first features of the environment are identified. The locations of these features are kept updated with the help of Kalman Filter assuming that the noise is Gaussian (Thrun, 2002; Milford, 2008, pp. 15-17). In order to use a Kalman Filter for mapping a kinematic model of the robot is required and the processes in the system have to be assumed to be linear. The Extended Kalman Filter remove this limitation of the standard Kalman Filter, so that non-linear processes can be used. One of the major challenges for Kalman Filter approaches to deal with two indistinguishable landmarks or features (Thrun, 2002). Such identical landmarks introduce multi-modal distributions over robot poses and could in essence lead to failure of the mapping algorithm (Thrun, 2002; Milford, 2008, pp. 15-17). Milford (2008) suggests that one of the ways to solve this problem is only consider unique features and landmarks, but adds that this would leave out many features of the environment and degrade the map. He additionally reports that the complexity of the Kalman Filter approaches increases quadratically with the number of landmarks and it becomes computationally expensive in mapping large environments. He says that the Sparse Extended Information Filter could be used for computational efficiency but there could degradation in the map built and lead to globally inconsistent maps.

2.2.3.2 Expectation Maximization approaches

The second approach to solve the map building problem, the Expectation Maximization (EM) approach. This approach consists of two steps. The expectation step is the one in which the posterior probability distribution over robot poses is determined for a given map while the maximization step is the one in which the algorithm “calculates the most likely map given these pose expectations” (Thrun, 2002). One of the greatest advantage of EM algorithms, as pointed out by Thrun (2002), is that they can solve the correspondence problem in mapping and generate accurate maps even in large cyclic environments. Thrun (2002) explains that the EM algorithm is able to do so as follows. First

several hypotheses about the robot's path are created in the expectation step. And each of the path, relates to different correspondences between environmental features and the data from the sensors. The correspondences create the features in the map in the maximization step. In the succeeding steps of the algorithm the correctly identified features of the environment are reinforced while the other features gradually disappear (Thrun, 2002; Milford, 2008, pp. 17-18). In spite of the above mentioned advantages, Milford (2008) mentions that the EM algorithms don't maintain a full notion of uncertainty and they can get stuck in the local maxima. He further adds that they are also computationally very expensive which makes it an offline algorithm.

2.2.3.3 Particle Filter approaches

The Particle Filter methods are another approach to solve the mapping problem. The Particle Filter method was initially introduced as a localization technique in which case it is also known as Monte Carlo Localization (Fox et al., 1999). In this method the probability of robot pose is represented by a samples or particles, which are used to approximate the posterior distribution over robot poses (Fox et al., 1999; Milford, 2008, p. 20). The advantage of particle filter is that it does not approximate the posterior probability distribution in parametric-form and hence it could be used in theory for any distribution. (Fox et al., 1999; Milford, 2008, p. 20). Also the properties like the number of particles, weighing of the particles can be easily adjusted to suit the problem at hand (Saarinen, 2009, p. 30; Milford, 2008, p. 20). According to Milford (2008), the Particle Filter methods have been extended to address the SLAM problem as well for e.g. the FASTSlam and FASTSlam 2.0 algorithms. He also says that the particle filter methods mentioned are computationally less expensive because the Extended Kalman Filter methods needs to maintain a five dimensional Gaussian distribution.

2.2.3.4 Scan Matching approaches

The Scan Matching approaches are another class of the methods that help in tackling the SLAM problem. Scan Matching is defined as "a process of calculating the differential movement on the basis of consecutive range scans"

by Saarinen (2009). He also mentions that the three basic methods of Scan Matching approaches are:

1. By searching in feature space
2. By searching in pose space
3. By using translation-invariant transformations with cross-correlation

He further elaborates the above methods as follows. In the first method, a search is made to find features like point or lines and a transformation is tried to find out that would best align the features. In the second method, a search is made in the pose space that would give the best correlation between successive scans or point-clouds. In the third method he says that the estimation of heading and translation is separated by transforming the scans or point-clouds into a function that is invariant to translation.

The Scan Matching method based on search in feature space has been implemented for this thesis using Iterative Closest Point (ICP) algorithm, which is explained in more in detail in Chapter 4.

2.3 Mapping in planetary rovers

The technology and algorithms used on autonomous robots on earth cannot be used directly for planetary rovers. These techniques and algorithms have to be adapted to suit the extreme environment and resource constraints on a planetary rover. The Mars Exploration Rovers (MER) had a passive stereo image processing system which was used to gather geometric information about the surrounding terrain (Maimone et al., 2006). This was carried out “by automatically matching and triangulating pixels from a pair of stereo-rectified images to generate a cloud of 3D points representing the imaged terrain” (Maimone et al., 2006). This stereo generated 3D geometric data would be used by Grid-based Estimation of Surface Traversability Applied to Local Terrain (GESTALT) system to build and maintain a grid-based local traversability map in rover memory (Maimone et al., 2006). This map, which is centred around the rover moves

along with the rover and it is updated as the rover moves and new images are obtained from the stereo vision system. This new traversability map is fused with older ones to obtain a global map of the surroundings (Maimone et al., 2006). Lacroix et al. (2002) have developed a method that creates a description of the terrain in the terms of navigability classes with the help of stereo-vision data to give qualitative information about traversability. The method classifies the terrain and makes a probabilistically labelled polygonal map, similar to an occupancy grid representation. The advantage of the method proposed by Lacroix et al. (2002) is that different view points can be fused into a global description, using the Bayes formula because of the probabilistic description (Lacroix et al., 2002). The other good sensor to model the surrounding environment of the planetary rover is the LIDAR. Ishigami et al. (2012) proposes a method to get a point cloud data of the terrain features using LIDAR. Various features of the terrain are embedded in the point cloud data given by the LIDAR. This is used by Ishigami et al. (2012) to make a digital elevation map with sector-shaped reference grid, “resulting in an elevation map with cylindrical coordinates” (Ishigami et al., 2012). Arras (2003) also uses features of the surrounding to make a map of it but it remains in 2D instead of 3D.

Once the map of the environment is obtained and the features extracted, the rover has to find a path devoid of hazards to reach the target location safely. Various algorithms like A^* , Dijkstra, D^* , potential field etc. have been developed to solve the problem of path planning, given a map. Ishigami et al. (2012) uses Dijkstra’s algorithm for path planning and the cost function in Dijkstra’s algorithm in his method include a combination of terrain inclination, terrain roughness, and path length. In the Mars Exploration Rovers (MER) *Spirit* and *Opportunity*, “the candidate motion paths are projected onto the traversability map, and a weighted evaluation of the constituent grid cells is assigned to each path.” (Maimone et al., 2006). The result is set consisting of Obstacle path evaluations in which low values indicate less traversable path and high values indicate high traversability. If there are several traversable paths, all of them are assigned Way-point path evaluations depending on the effectiveness of each path to drive the rover to the goal. The path leading directly towards the goal point is given highest evaluation and other parts are given values which are less according to a Gaussian distribution (Maimone et al., 2006).

Localization plays an important role while mapping. Wheel odometry would not be very useful for localization on planetary rovers because of unbounded errors in presence of wheel slippage and rough terrain. The MERs utilize stereo camera images to correct the errors accumulated in position estimated by wheel odometry. They achieve this by “comparing locations of features found in stereo image pairs taken before and after a small motion step” (Maimone et al., 2006). The MERs also had the capability of Visual Odometry which gave a pose update “by tracking the motion of interesting terrain features between two pairs of stereo images in both 2D pixel coordinates and 3D world coordinates. A maximum likelihood estimator was applied to the computed 3D offsets between features in successive images to produce the final motion estimate” (Maimone et al., 2006).

Although LIDAR scanners have not yet been used on a actual planetary rover, they provide a promising alternative to the stereo cameras for map building. The 3D LIDAR scanners would enable the rover to build accurate long range maps. From the algorithmic point of view, the scan-matching approach to map building is simple in concept to be used to build maps of desert like environments of the Moon. The rigid transformation between two terrain scans could be used to correct the odometry information. The Iterative Closest Point (ICP) algorithm could be used to combine the various terrain scans together to build a map and provide the odometry corrections. Gemme et al. (2012) have used the ICP algorithm for mapping and pose correction on a Martian rover prototype in the Mars Emulation Terrain (MET) of the Canadian Space Agency (CSA). They were able to maintain the localization error of 1% for traversal of 102.6 meters.

Chapter 3

Implementation

3.1 Hardware platforms and software

3.1.1 Hardware platforms

3.1.1.1 Mobile platform

The mobile platform used for the experiments was the *Pioneer 3-DX* from Adept MobileRobots, Inc. The *Pioneer 3-DX* is a differential drive robot, which comes with sensors like sonar and wheel encoders (Pioneer, 2013). The *Pioneer 3-DX* robot interfaces quickly with Robot Operating System (ROS), which is the meta operating system used for the thesis. Figure 3.1 shows the *Pioneer 3-DX* robot.



Figure 3.1: Pioneer 3DX. Image from Pioneer (2013)

3.1.1.2 LIDAR scanner

The LIDAR scanner used was the LMS100 from *SICK AG* which is a 2D LIDAR scanner. The LMS100 is interfaced with the computer via Ethernet. In order to get a point-cloud from the 2D LIDAR scanner a tilting mechanism was built to tilt the LIDAR scanner back and forth along the pitch axis. The Figure 3.2 shows the axis along which the LIDAR scanner was pitched along with the resulting point-cloud.

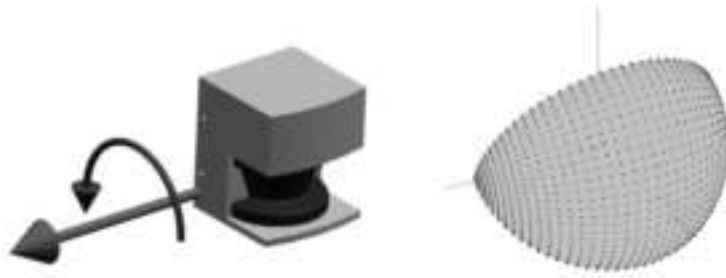


Figure 3.2: Pitching scan and the resulting cloud. Figure from Wulf and Wagner (2003)

3.1.1.3 Micro-controller and servo

For tilting the LIDAR scanner, a Futaba-S5301 servo controlled by a *Arduino Mega 2560* micro-controller board was used. The *Arduino Mega 2560* is a micro-controller board based on the ATmega2560 and has all the peripherals of typical micro-controller like Universal Asynchronous Receiver/Transmitters (UARTs), Timers, Pulse Width Modulation (PWM) outputs etc. (Arduino, 2013). The Figure 3.3 shows the servo and the *Arduino* board.

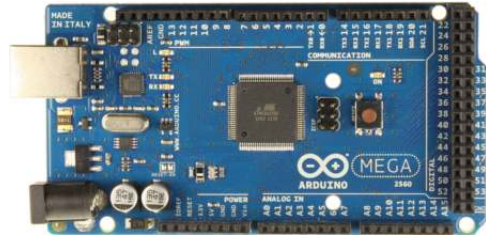
3.1.2 Software

3.1.2.1 Robot Operating System

The software used for the implementation was Robot Operating System (ROS). Although, ROS stands for Robot Operating System (ROS), ROS is not an operating system which does the work of process management and



(a) Futaba S5301 high torque servo



(b) Arduino board. Image from Arduino (2013)

Figure 3.3: Servo and micro-controller

scheduling; “rather, it provides a structured communications layer above the host operating systems of a heterogeneous computer cluster.” (Quigley et al., 2009). As mentioned in Quigley et al. (2009), ROS has the following design characteristics.

- *Peer-to-peer* A robot built using ROS, has several processes running on different machines which are connected to each other by peer-to-peer technology. ROS provides a look-up methodology called “master”, for the processes to locate each other
- *Multi-lingual* ROS is designed to be language neutral and presently supports C++, Python, Octave and LISP. ROS uses language-neutral interface definition language to describes the messages between various parts. The negotiation and configuration for peer-to-peer communication is done with the help of XML-RPC.
- *Tools-based* ROS has a microkernel design, which consists of various small software tools for building and running the different ROS modules.
- *Thin* The build systems in ROS, build the source code tree in a modular way using CMake. ROS design encourages to place all the complexities of the software in libraries and creates only small executables which exposes library functionality to ROS. This is said to help in easier code extraction and reuse besides the advantage of easier unit testing.
- *Free and Open Source* ROS has been licensed under the permissive BSD license which allows it to be used for both commercial and non-commercial purposes. ROS source code is also publicly available for free.

It is important to understand the following main concepts which are fundamental to the implementation of ROS. These terms which are explained in Quigley et al. (2009), are mentioned here in brief.

- *Nodes* Node is one fundamental unit performing computation and can be regarded as software module. The communication between different nodes is managed by a master node called `roscore`
- *Messages* A message is strictly typed data structure that is used for communication between different nodes. ROS messages supports standard primitive data types like the integers, floats, characters and messages which are made up of other messages.
- *Topics* ROS Messages are sent on a particular Topic by a node. A node can also receive messages from a topic. The above two processes are called publishing and subscribing respectively. A node is allowed to publish and subscribe on many topics.
- *Services* A ROS service provides a response when a request is made to the service. This request/response model is similar to the request/response model of web-services. A service is advertised with a particular name. A node can ask for this service by sending a request on this particular name and would receive a response for its request.

3.1.2.2 Point Cloud Library

“A point-cloud is a data structure used to represent a collection of multi-dimensional points and is commonly used to represent three-dimensional data” (PCL-Documentation, 2013c). When the sensor gives out data that has only three dimensions namely the X, Y and Z axis then the point-cloud is of 3 dimensions. However, when the sensors also additional information like colour then the point-cloud has 4 dimensions (PCL-Documentation, 2013c). The Microsoft Kinect is an example of a sensor that provides data with 4 dimensions.

“The Point Cloud Library (PCL) is a comprehensive free, BSD licensed, library for n-D Point Clouds and 3D geometry processing” (Rusu and Cousins,

2011). The PCL is a templated C++ library that consists of many state-of-the-art algorithms for 3D/n-D point-cloud processing which includes filtering out noise, segmentation of the data into various objects, registration of several point-clouds and visualization of the n-D point-clouds to name a few (PCL-Documentation, 2013c).

3.2 System overview

The overall system of hardware and software is controlled by the interaction of various nodes. On a broad level, the Figure 3.4 on the following page shows the main nodes that make up the system. The user interacts via the `operator_node` called `Rob_key` to give commands. The user can drive the mobile base around and give commands to get terrain scans with the help of the `operator_node`. The `Mobile Base Driver node` takes commands from the `operator_node` and sets the velocity and pose of the mobile base. It also publishes the odometry, which is used by the `ICP registration node` as an initial guess for the algorithm. The `angle_publisher` node waits for scan commands from the `operator_node`. When the `angle_publisher` node receives the scan command from `operator_node`, it sends out the angles at which the LIDAR scanner should be, to the `arduino_node`. The `arduino_node` written inside the arduino micro-controller board receives these angles and positions the LIDAR scanner at the appropriate angle. A terrain scan consists of tilting the LIDAR scanner from the horizontal position to the tilted position and then back again to the horizontal position. The angles at which the 2D LIDAR scanner is tilted is also published to the `laser_assembler` node. The `laser_assembler` node keeps a rolling buffer of 400 2D LIDAR scans with the angles at which the 2D LIDAR scans were taken. The job of the `laser_assembler` node is to combine the two-dimensional LIDAR scans into point-clouds. The `laser_assembler` node provides this point-cloud as a service. The `angle_publisher` node makes a note of the start time and end time of the scan and then asks the `laser_assembler` node to combine the 2D LIDAR scans between start time and end time into a point-cloud. The `laser_assembler` node returns the point-cloud to the `angle_publisher` node which in turn publishes this point-cloud to the `ICP registration node`. The `ICP registration node` takes this point-cloud and the odometry information and makes a 3D map.

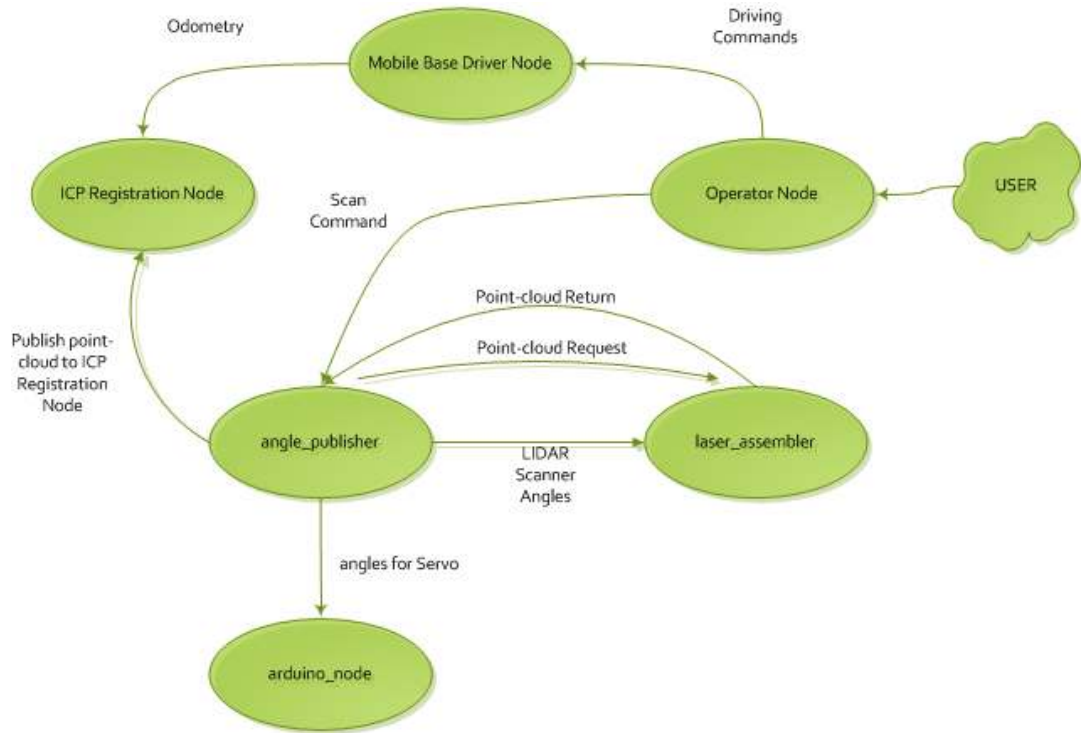


Figure 3.4: Broad overview of the software nodes.

A detailed node graph of the system is shown in Figure 3.5 on the next page. The ovals in the figures are the nodes and the rectangles are the topics used by the nodes. The `Rob_key` is the `operator_node`. The topics `rosout` and `ros_agg` are related to the master node which helps in communications between the various nodes. The `robot_state_publisher` node helps in keeping track of the position of the various joints in the system. The node `LMS_100` is the driver for the LIDAR scanner while the `Laser_filter_node` filters the data from the driver node.

3.2.1 ROS stacks/packages used in the system

The packages used in the system built are described here in brief.

- *tf*: This package helps in keeping track of various coordinate frames the robots uses over time (ROS-Wiki-tf, 2013). It does so by storing the relationship between the various coordinate frames in a tree structure buffered in time (ROS-Wiki-tf, 2013).

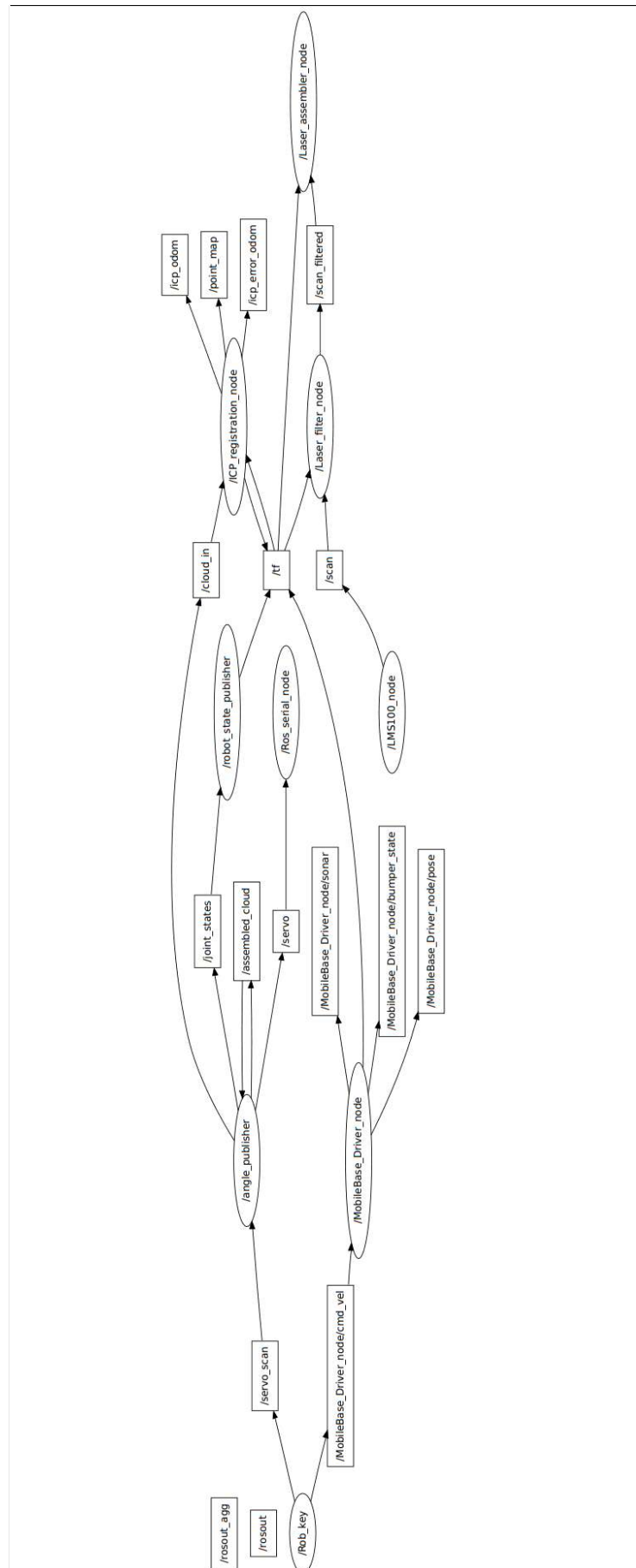


Figure 3.5: Node graph of the system

- *robot_state_publisher*: This package helps in broadcasting the state of the robot to all nodes using `tf` information. The package takes in data about the joint angles of the various joints in the robot and publishes it as `tf` data, by using a kinematic tree model of the robot written in Unified Robot Description Format (URDF) (ROS-Wiki-RST, 2013).
- *Rosaria*: Advanced Robot Interface for Applications (ARIA) is a C++ library provided by Adept MobileRobots, Inc. for their robot platforms like the Pioneer-3-DX. ARIA helps in tele-operating the mobile platform besides providing information about from odometer and other sensors on the mobile platform (Adept-MobileRobots, 2013). `ROSARIA` is the ROS wrapper package for the ARIA library, that helps in using ARIA easily in ROS.
- *LMS1xx*: The `LMS1xx` is driver package for the SICK LMS1xx series of LIDAR scanners. It takes the sensor output and converts it into a ROS compatible message.
- *laser_assembler*: This package helps in converting data from an articulating LIDAR scanner into a 3D point-cloud (ROS-Wiki-laser-assembler, 2013). The node that provides this service is called the `laser_scan_assembler`. The working of the node is shown in the Figure 3.6 on the following page. Whenever a 2D laser scan comes in, the node transforms it into the 3D Cartesian space with the help of `tf` data. The node keeps a rolling buffer of 400 scans. Another node can ask for a point-cloud between specific time from the `laser_scan_assembler` node.
- *laser_filters*: The `laser_filters` package provides general purpose filters for processing data from 2D LIDAR scanners (ROS-Wiki-laserfilters, 2013). The angular bounds filters from this package was used to reduce the field-of-view of the LIDAR scanner from 270° to 140° in the yaw axis as explained in section 4.3 on page 41.
- *roserial*: The `roserial` stack provides a protocol for transferring and receiving ROS messages over serial links (Ros-Wiki-roserial, 2013). The `serial_node.py` node from the `roserial_python` package was run on the host machine to connect to the node running on the *arduino* board. This stack was used to send the angles to which the servo had to set.

- *ethzasl_icp_mapping*: This stack helps in making 3D maps from point-clouds. It provides a SLAM system based on the ICP algorithm explained in Chapter 4.

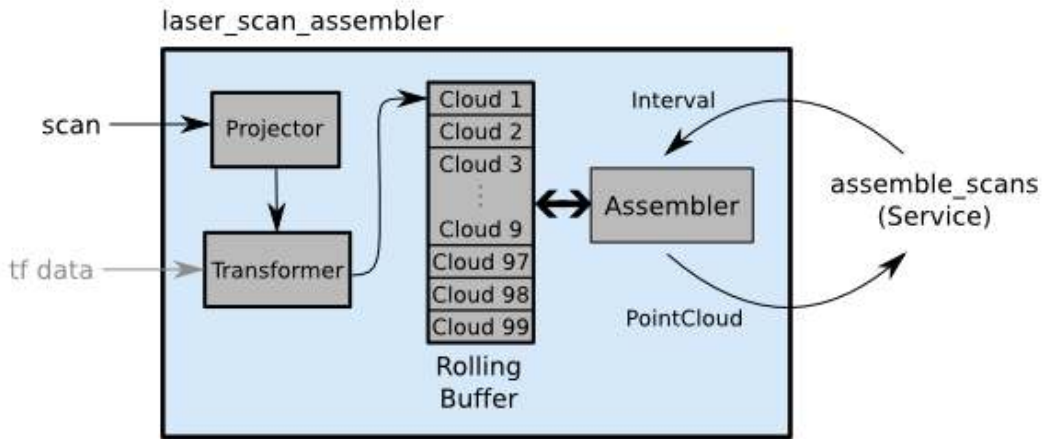


Figure 3.6: Working of the `laser_scan_assembler` node. Figure from ROS-Wiki-laser-assembler (2013)

3.3 Tilting system

In order to obtain a 3D point-cloud with the help of a LIDAR scanner, it has to be articulated in the roll, pitch or yaw axis. In our case, it was decided to tilt the LIDAR in the pitch axis because obstacles above the horizontal are not useful for terrain mapping. For accomplishing this tilting, on the pitch axis of the laser, it was connected with a futaba high torque servo with a four-bar linkage mechanism. The tilting system consisted mainly two parts, the electronic control of the servo and the four-bar linkage mechanism. The mechanical design of the four-bar linkage was done by Tomi Ylikorpi and was manufactured by Matti Lassila from The Finnish School of Watch-making(Micro-mechanics programme).

3.3.1 Servo control with ROS and Arduino

The main ROS packages used for servo control are the `rosserial_python` and `rosserial_arduino` packages. The `serial_node.py` ROS node in the

`rosserial_python` package allows serial communications between micro-controllers and the computer running the `roscore` master node. The `rosserial_arduino` packages helps in making ROS nodes in the micro-controller to which commands can be issued. The hardware connection of the servo with the arduino board is shown in Figure 3.7.

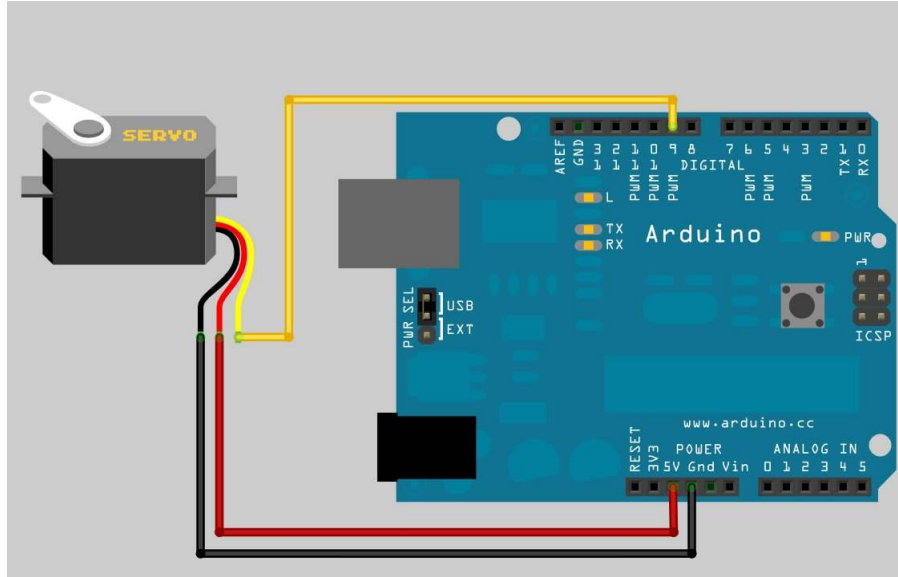


Figure 3.7: Arduino connection with servo. Figure from ROS-Wiki-Arduino (2013)

The *Arduino* board is connected to the computer running `roscore` master node with a Universal Serial Bus (USB) cable. The arduino board has an on-chip serial to USB converter. This brings up the *Arduino* board as a serial port on the computer. This virtual serial port can be used by the `rosserial_python` package to connect to the *Arduino* board. The flow diagram of the `arduino_node` for using ROS node on the *Arduino* board is given in Appendix B on page V. The motion of the servo can be used to tilt the LIDAR scanner in its pitch axis.

3.3.2 Four-bar link mechanism

In order to convert the circular motion of the servo into the tilting motion of the LIDAR scanner a mechanism is needed. A four-bar crank-rocker mechanism is a good way to achieve the above objective. The Figure 3.8 shows a simple

crank-rocker mechanism in two different positions. The segment AB need not be an actual link. It is called the Ground Link because it makes no motion and the points A and B are fixed. The segment AF' is the crank and the point F' moves on the circle shown. The segment BG is the rocker that makes back and forth motion. The connecting segment $F'G$ is called the coupler.

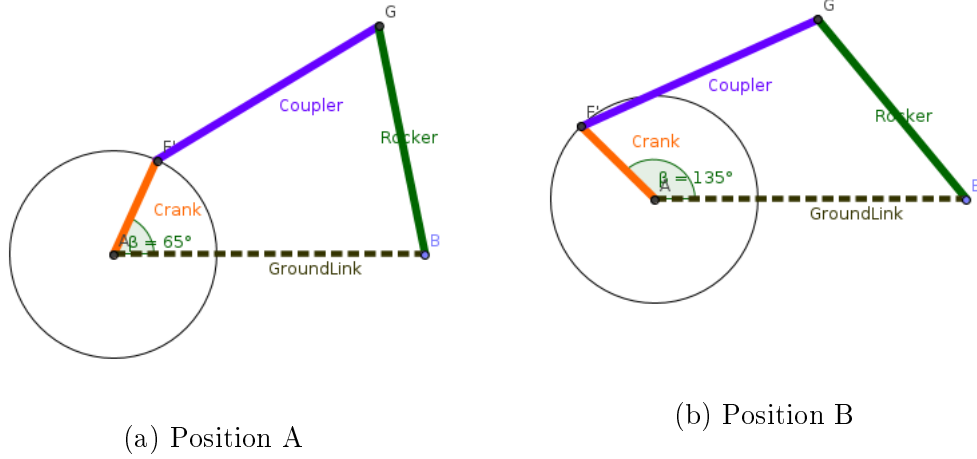


Figure 3.8: Four-bar crank-rocker mechanism in two different positions

If the crank is connected to the servo and LIDAR scanner is pivoted on the point B , it would be possible to tilt the LIDAR scanner back and forth. The Figure 3.9 on the following page shows a generalized crank-rocker configuration that can be used to find the relationship between the position of the crank and the position of the rocker.

Let the lengths of the four links be r_1 , r_2 , r_3 and r_4 as shown in Figure 3.9 on the next page. Let γ be the angle between the ground-link and the horizontal, α be the angle between the crank and the horizontal and β be the angle between the rocker and the horizontal. It can be shown that relationship between β and α is

$$\beta = \tan^{-1} \left(\frac{y}{x} \right) + \cos^{-1} \left(\frac{x^2 + y^2 + R_3^2 + R_4^2}{2R_4\sqrt{x^2 + y^2}} \right) \quad (3.1)$$

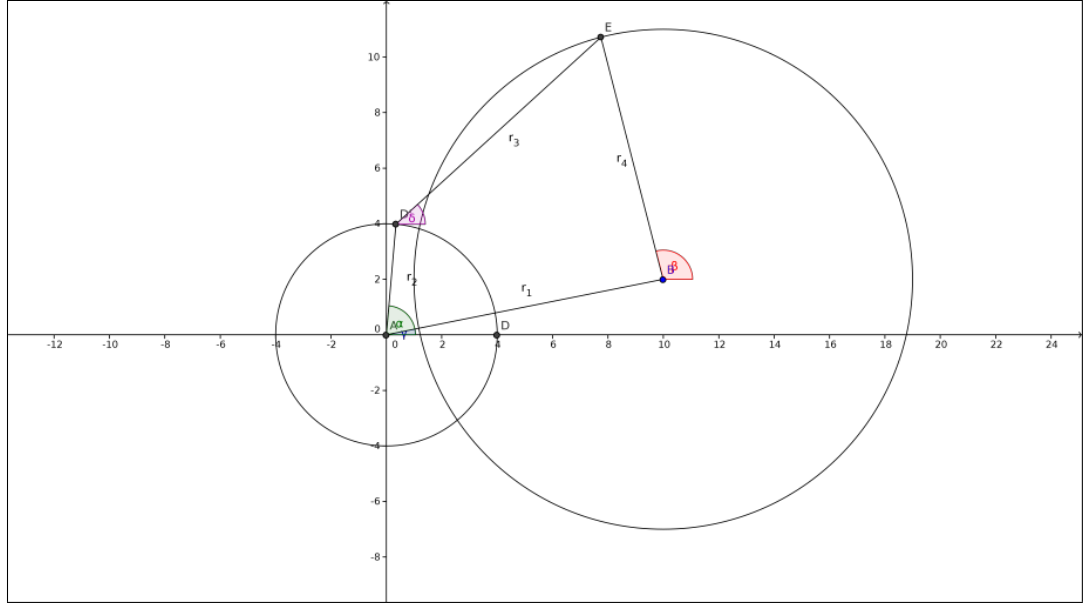


Figure 3.9: Generalized crank-rocker four-bar linkage

where

$$x = r_1 \cos(\gamma) - r_2 \cos(\alpha) \quad (3.2a)$$

$$y = r_1 \sin(\gamma) - r_2 \sin(\alpha) \quad (3.2b)$$

$$R_3 = r_3 \quad (3.2c)$$

$$R_4 = -r_4 \quad (3.2d)$$

as proved in (SoftIntegrationInc., 2013). The complete proof with the notations in Figure 3.9 is given in Appendix A on page I. The Figures 3.10 and 3.11 show the geometrical diagrams of the tilting mechanism that was constructed. They show the mechanism in the horizontal and tilted positions. The LIDAR is pivoted at point B. The segment AB acts as the ground-link. The Figure 3.12 shows the actual photographs of the mechanism in the two positions corresponding to Figure 3.10 and 3.11.

3.4 Assembling laser-scans to point-clouds

In order to convert the laser-scans from the tilting LIDAR scanner into a point-cloud, the angle of the LIDAR scanner after each scan needs to be broadcast or published so that the node `laser_assembler` can combine the laser-scans into a point-cloud. A URDF model of the mobile platform was used to

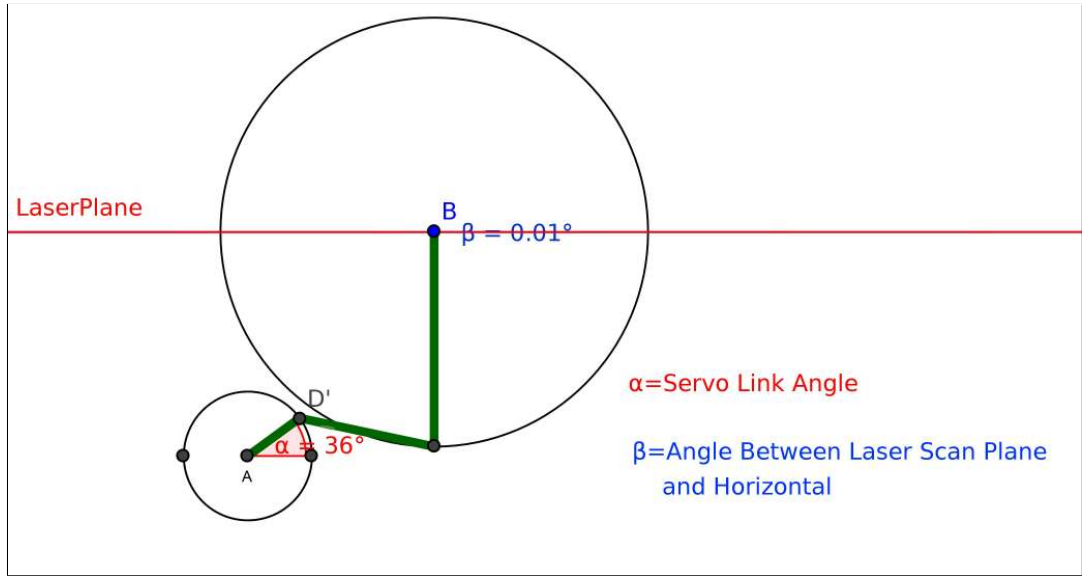


Figure 3.10: The mechanism showing horizontal position of LIDAR scan-plane

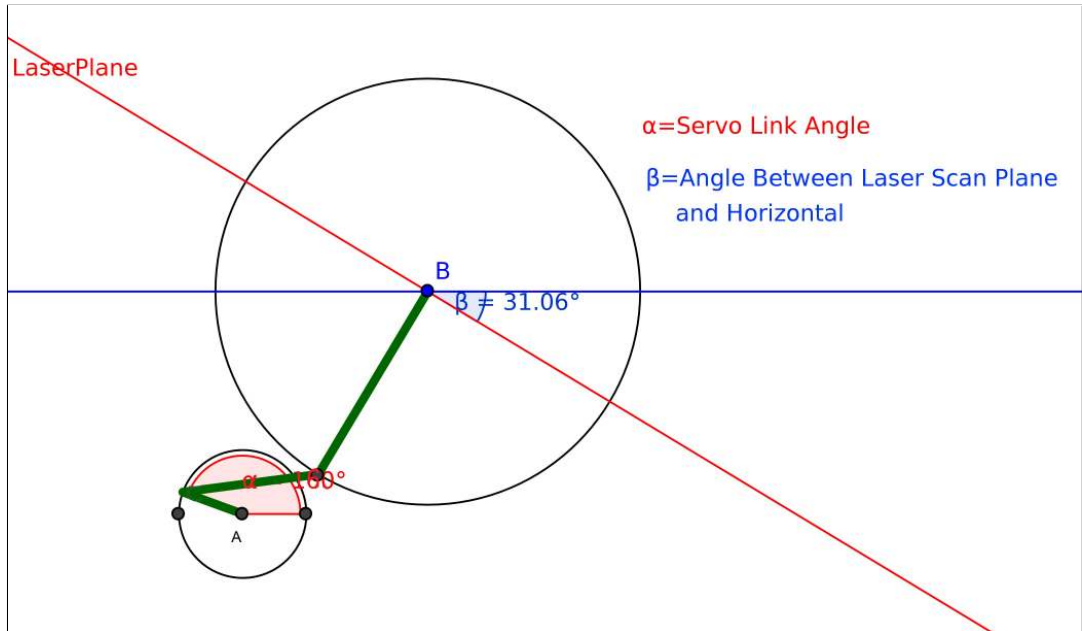


Figure 3.11: The mechanism showing tilted position of LIDAR scan-plane

achieve the above objective. URDF stands for *Unified Robot Description Format*. It is used in ROS to describe a robot. URDF is written in XML to describe the various parts of the robot like sensors, the relation between various joints and link and their kinematic and dynamic properties, kinematic and dynamic properties of the whole robot and others (ROS-Wiki-URDF, 2013). A simple URDF model of the robot was created describing the relationship between the centre of the mobile base and the laser. A frame called `base_link` is attached



(a) The mechanism in horizontal position (b) The mechanism in tilted position

Figure 3.12: Photos of the actual mechanism built

to the centre of the mobile base and a frame called `laser` is attached to the laser. The URDF description of the mobile platform with the laser on it is given in listing 3.1 on the next page.

The first line is the name of the XML file. The second and third lines create two frames with the name `base_link` and `laser`. From the fifth line onwards the joint connecting the two frames or links is created and its properties defined. The eighth line describes the location of the joint connecting the two frames in terms of “xyz” translation and “roll, pitch, yaw” rotations. The parameter axis on the ninth line describes the axis along which the joint is allowed to rotate, i.e. the pitch axis or y-axis.

The `laser_assembler` package requires the angle of the laser frame as a `tf` message between the `base_link` frame and `laser` frame. The package `robot_state_publisher` helps in converting the joint angles into a `tf` message. When a change is made to the angle of the LIDAR scanner with the horizontal, it has to be published on the `joint_states` topic. The `robot_state_publisher` takes this messages on the `joint_states` topic and publishes a `tf` message between the `base_link` frame and `laser` frame. The `angle_publisher` node helps in coordinating the commanding of the servo, broadcasting the angle of the laser joint, service call for point-cloud from the `laser_assembler` node and

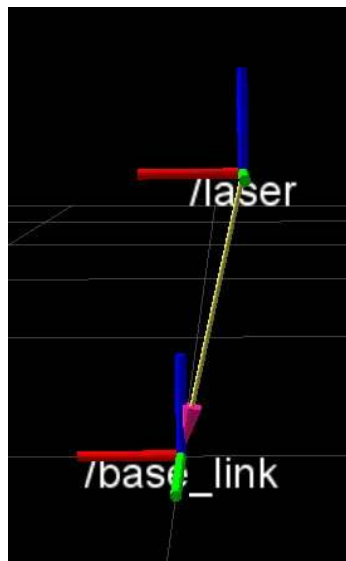
finally publishing this point-cloud to the ICP Registration node. The flow diagram of the `angle_publisher` node is given in Appendix B on page V.

Listing 3.1: URDF for the platform used

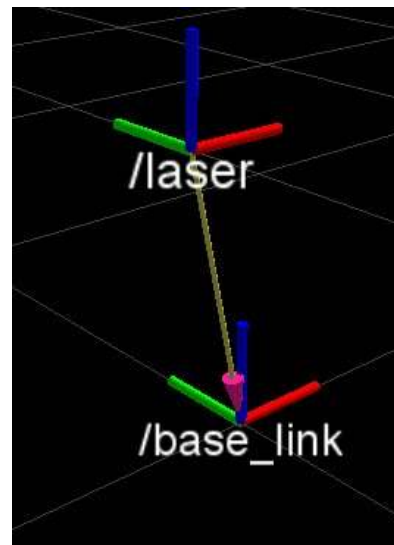
```

1 <robot name="anuraj_pioneer">
2   <link name="base_link" />
3   <link name="laser" />
4
5   <joint name="laser_joint" type="continuous">
6     <parent link="base_link"/>
7     <child link="laser" />
8     <origin xyz="-0.12 0.0 0.55" rpy="0 0 0" />
9     <axis xyz="-0 1 0" />
10  </joint>
11 </robot>

```

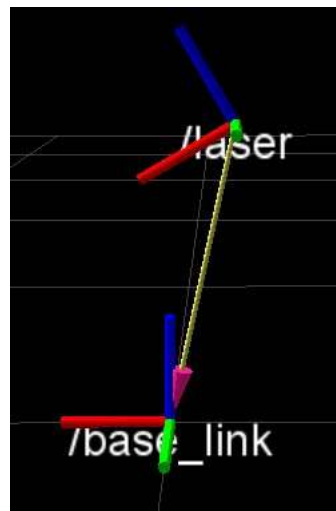


(a) View A

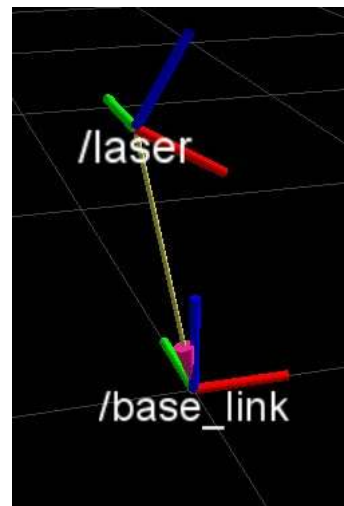


(b) View B

Figure 3.13: The URDF model visualized with laser in horizontal position



(a) View A



(b) View B

Figure 3.14: The URDF model visualized with laser in tilted position

Chapter 4

Global map building with ICP-based SLAM

Once the terrain scans are available as point-clouds, the next step to build a map is to combine the point-clouds into a common co-ordinate system. This process of combining the point-clouds into a common co-ordinate system is called registration and the Iterative Closest Point (ICP) algorithm is one of the popular algorithms for the registration of 3D point-clouds (Nüchter, 2009). The ICP algorithm was independently invented by Besl and McKay (1992) and Chen and Medioni (1992) in the early 1990s (Nüchter, 2009).

4.1 The Iterative Closest Point algorithm

The ICP algorithm helps in combining the point-clouds from the 3D LIDAR scans into a common co-ordinate system. Let us suppose that there are two 3D point sets of the same object which have been acquired with the help of the 3D LIDAR scanner and have sufficient overlapping features. Let the first set be called target set, denoted by \hat{M} and the second set be called source set, denoted by \hat{D} . The objective of the algorithm is to align the source point-cloud to the target point-cloud. Let the number of points in \hat{M} be N_m i.e. $|\hat{M}| = N_m$. Let the number of points in \hat{D} be N_d i.e. $|\hat{D}| = N_d$. The tuple (m_i, d_j) are corresponding points from the point-sets \hat{M} and \hat{D} respectively. Nüchter (2009) points out that the ICP algorithm helps in finding a transformation matrix

comprising of a rotation matrix \mathbf{R} and a translation matrix \mathbf{t} , which would try to minimize the error function:

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \| \hat{\mathbf{m}}_i - (\mathbf{R}\hat{\mathbf{d}}_j + \mathbf{t}) \|^2 \quad (4.1)$$

Algorithm 1 The ICP algorithm from Nüchter (2009)

- 1: **for** $i = 0$ *to* $maxIterations$ **do**
 - 2: **for all** $\mathbf{d} \in D$ **do**
 - 3: find closest point within a range d_{max} in the set M for point \mathbf{d}_j
 - 4: **end for**
 - 5: Calculate transformation (\mathbf{R}, \mathbf{t}) that minimizes the error function in Eq 4.1.
 - 6: Apply transformation found in Step 5 to the data set D
 - 7: Compute the difference of the quadratic error, i.e. compute the difference of the value $\| E_{i-1}(\mathbf{R}, \mathbf{t}) - E_i(\mathbf{R}, \mathbf{t}) \|$ before and after the application of the transformation. If this difference falls below a threshold ϵ , terminate.
 - 8: **end for**
-

Nüchter (2009) further simplifies Equation 4.1 to:

$$E(\mathbf{R}, \mathbf{t}) = \frac{1}{N} \sum_{i=1}^N \| \mathbf{m}_i - (\mathbf{R}\mathbf{d}_i + \mathbf{t}) \|^2, \text{ using } N = \sum_{i=1}^{N_d} sgn(w_{i,j}) \quad (4.2)$$

He mentions that the simplified version of the error function given in Equation 4.2 is the one generally used for implementation. The odometry from the mobile platform is used as an initial guess for calculating the transformation in step 5 of Algorithm 1. The ICP algorithm would theoretically work without initial guesses, but successive scans would require large overlapping regions. This in turn would reduce the traversing speed of the rover. The initial guess need not be from the wheel odometry. It could be from any sensor estimating the pose of the rover, for e.g. stereo camera by visual odometry or by using a star tracker or by using multiple sensors. The iterative steps involved in ICP alignment method is given in Algorithm 1. For implementing the ICP algorithm in

the context of the thesis, the modular ICP library `libpointmatcher` was used (Pomerleau et al., 2013; Pomerleau et al., 2011).

ICP is a popular algorithm for 3D robotic mapping due to its simple underlying concept. A large number of variants for ICP have been proposed during the last two decades for different applications. But all these variants have two main steps in common (Segal et al., 2009)

1. finding out corresponding points between successive scans
2. calculating the transform that would minimize an error function between the corresponding points.

4.1.1 Finding closest point using kD-tree

The step of finding out corresponding points based on the closest point criterion heavily affects the performance of the ICP algorithm. The determination of closest points has to be done quickly to reduce the computational time of the ICP algorithm (Nüchter, 2009, p. 52). The most simple way of finding out the closest point in the two point-sets is by linearly searching through the points in them. This method is not used in practice because the complexity of this method scales with $\mathcal{O}(n^2)$, where n is the number of corresponding points in the two point-sets. This step of searching the closest point can be completed faster by building a k-dimensional(kD) tree data structure from the point-sets. By building a kD-tree data structure the search complexity is reduced to $\mathcal{O}(n \log n)$ (Pomerleau, 2013, p. 26). In the case of the 3D point-set the kD-tree is of order 3.

In a kD-tree data structure, which is a variant of binary search tree, each node divides the point-set into two unique sets along one of the principal axis and the root of the tree represents the whole point-set (Nüchter, 2009, p. 53). A kD-tree for a 2D data set is shown in Figure 4.1 on the next page. The first split is made at the point 1 along the vertical axis dividing the point-set into two. Then splits are made at the point 2 and point 3 along the horizontal axis.

These splits continue alternatively along the two axes. For a point-set in three dimensions the splits alternate in a round-robin fashion along the three axis.

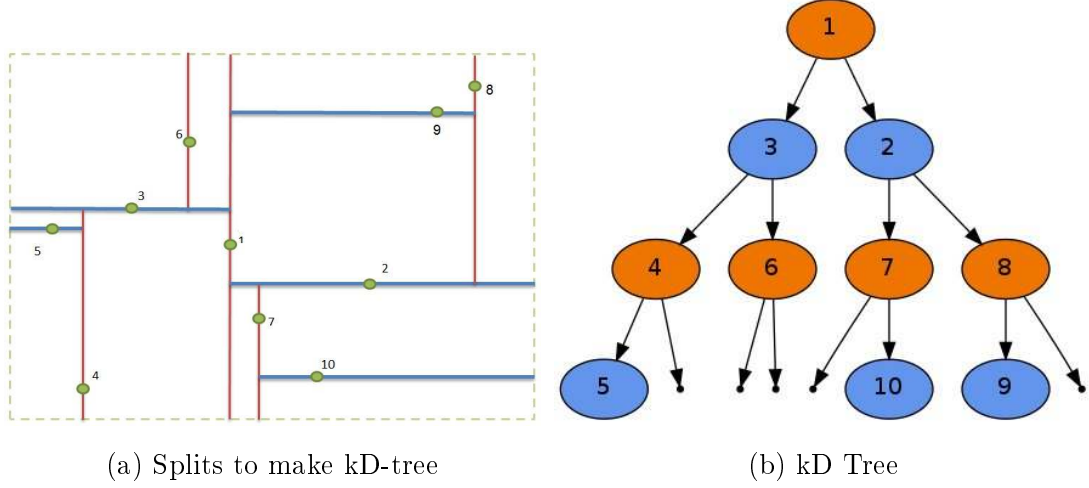


Figure 4.1: kD-tree of the order 2. Figure adapted from Sedgewick and Wayne (2013)

4.1.2 Error minimization

The choice of the error function also affects the performance of the ICP algorithm. The two classic error functions widely used for ICP are: a) the point-to-point error metric and b) point-to-plane error metric. The point-to-point error metric was used by Besl and McKay (1992), while the point-to-plane error metric was used by Chen and Medioni (1992) in their seminal work which introduced ICP. Low (2004) points out that the point-to-point error metric is the one in which “the sum of the squared distance between points in each correspondence pair is minimized”. While in the point-to-plane error metric, he says that “the object of minimization is the sum of the squared distance between a point and the tangent plane at its correspondence point.” The Figure 4.2 on the following page explains graphically the two classic error metrics used in ICP algorithm. The Figure 4.2 shows the cross-section of the two surface point-sets. The points p and q are the corresponding points. The distance denoted by d_s is the one which is minimized in the two different metrics. Pomerleau et al. (2013) argue that despite the fact that the point-to-plane error metric being superior in general, it could be less precise when there are large disturbance in the initial alignments and it could lose the superiority in unstructured environments.

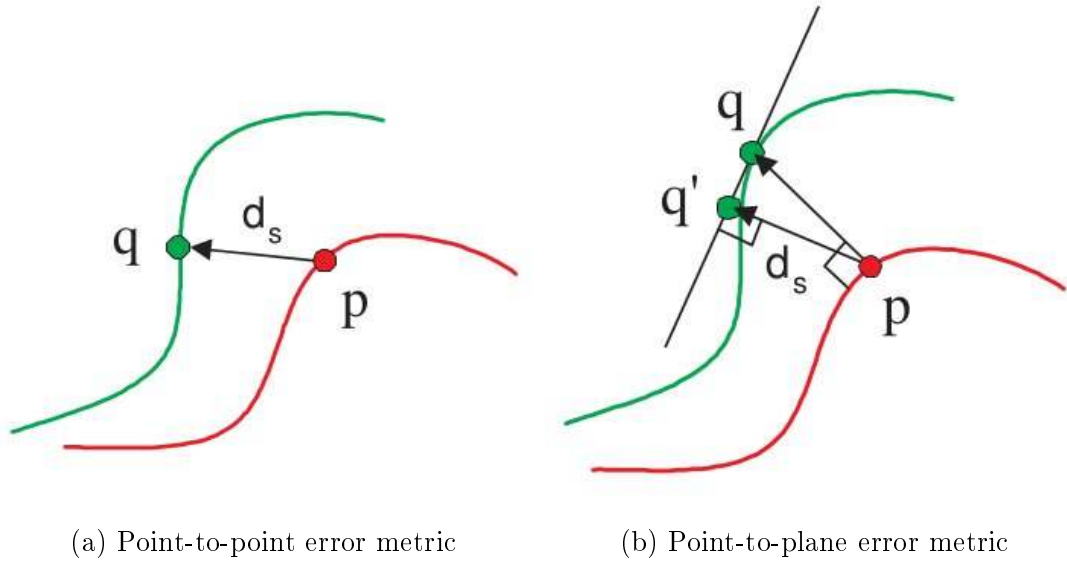


Figure 4.2: Classic error metrics for ICP. Figures from Park and Subbarao (2003)

4.2 Map representations

Once the scans are combined into a common co-ordinate system the resulting map is made up of point-clouds. The rover would require conversion of this point-cloud map into a suitable representation for navigation decisions. Hornung et al. (2013) argue that a good map representation needs to have the following features:

- *Probabilistic representation*: The information about the environment from the 3D sensors is not accurate and contains noise. These uncertainties and noises have to be dealt probabilistically. Multiple uncertain information could be combined to form an accurate representation of the environment.
- *Modelling of unmapped areas*: For planetary exploration this factor becomes even more important. The map should also represent free areas so that proper actions can be taken to map the unmapped areas.
- *Efficient representation*: This feature is also quite important because it plays a role when navigation decision are taken and are being executed. The map representation needs to be efficient because of the resource constraints on the rovers. If the memory requirement is large then it slows

the exploration speed of the rover and it could become a bottleneck for the rover system.

The commonly used map representations are point-clouds, elevation maps, voxel-grids and octree maps (Hornung et al., 2013). Point-cloud map is the one which we obtain directly from our implementation. The environment is represented by a cloud of points each having a x, y and z coordinate. The occupancy grid mapping in 2D could be extended to 3D by discretizing the environment into a grid of cubic volumes, known as voxel-grids (Hornung et al., 2013). The elevation maps also known as 2.5D maps, contain a z value or height for the 2D occupancy grid. Octree maps are based the octree data structure wherein the map is recursively divided in octants when required. Hornung et al. (2013) have proposed an implementation of octree representation called the *Octomap* framework which has been used in thesis work. The *Octomap* framework was used to create octree maps for three different scenarios and they are shown in chapter 5. The various types of map representations are shown in Figure 4.3. The advantages and disadvantages of the different map representations are given in Table 4.1 on the following page.

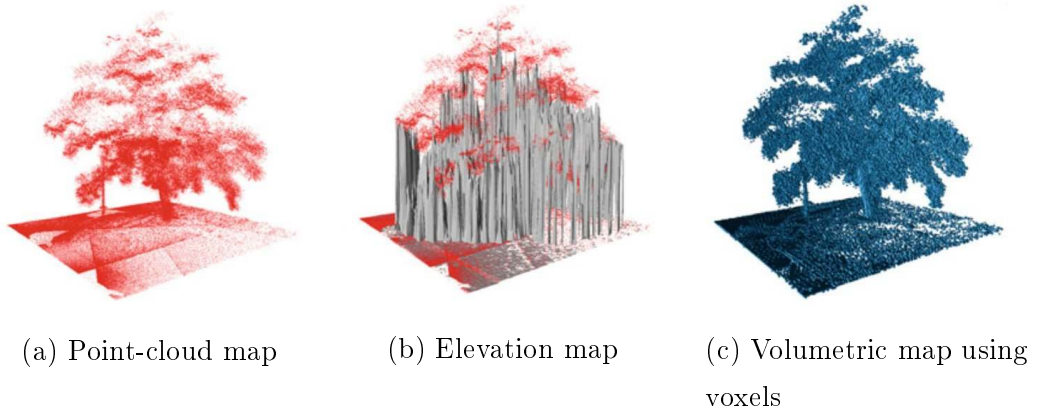


Figure 4.3: Different types of map representation. Figures from Hornung et al. (2013)

4.3 Data filtering

The sensors that are used in robotic mapping in general add noise to the output data they produce. The LIDAR scanner also suffers from this prob-

Table 4.1: Advantages and disadvantages of different map representations (Hornung et al., 2013)

Map representation	Advantage	Disadvantage
Point-cloud	<ul style="list-style-type: none"> • Directly obtained from sensors 	<ul style="list-style-type: none"> • No Modelling of free or unknown space • No handling of sensor noise • No upper limit on memory consumption
Elevation maps	<ul style="list-style-type: none"> • Simple implementation • Memory Efficient 	<ul style="list-style-type: none"> • Map does not represent real environment • Not Probabilistic
Voxel-grids	<ul style="list-style-type: none"> • Probabilistic update • Direct Discretization of environment 	<ul style="list-style-type: none"> • Requires initialization of map • Requires large memory
Octree maps	<ul style="list-style-type: none"> • Probabilistic • Represent real environment in 3D • Memory Efficient 	<ul style="list-style-type: none"> • Implementation is difficult

lem. Moreover, many times only a certain part of the sensor data output is required for the application at hand. Filtering of the sensor output data helps in reducing noise besides being useful in narrowing down the sensor data output to useful range. Filtering also helps in sub-sampling in order to reduce the number of points in the point-set. This helps in decreasing the computation time of the algorithm. The filters applied to the point-cloud data can be divided primarily into two types: 1) Filter applied before registration 2) Filter applied after registration.

The pre-registration filters were mainly used to decrease the number of redundant points. The filters that were applied before registration were as follows:

- *LIDAR Scanner Angular Bounds Filter*: The angular range of the field-of-view of the LIDAR scanner LMS 100 is 270° i.e. from -135° to $+135^\circ$ in the yaw axis. The LIDAR scanner was supposed to get scans of the surrounding terrain. The large angular field-of-view of the LIDAR scanner had to be reduced to 140° i.e. from -70° to $+70^\circ$ for the above reason. A laser scan angular bounds filter from the `laser_filter` package in ROS was used for this purpose.
- *Random Sampling Data Points Filter*: This filter is based on the ICP variant proposed by Masuda et al. (1996) with robust outliers rejection using random sampling. So this filter reduces the number of points in the point-cloud by random sampling (Magnenat, 2013). This filter is available in the stack `ethzasl_icp_mapping`
- *Trimmed Distance Outlier Filter*: This filter is based on the ICP variant proposed by Chetverikov et al. (2002) known as Trimmed ICP. This ICP variant is based on the *Least Trimmed Squares* method. So in this filter a specific percentage of corresponding points with the smallest norm are considered as inliers (Magnenat, 2013). This filter is also a part of the stack `ethzasl_icp_mapping`

The post-registration filters were applied to remove the outlier data and reduce the number of redundant data points. The filters that were applied after point-cloud registration were as follows:

- *Maximum Density Data Points Filter:* This filter was used for reducing the number of points in the point-clouds. This filter reduces the number of points by removing points at random when the density gets higher than a predefined threshold (Magenat, 2013). This filter is a part of the stack `ethzasl_icp_mapping`
- *Maximum Point Count Data Filter:* This filter was also used to reduce the number of points in the point-cloud. It tries to reduce the number of points in the point-cloud by removing points at random if the total number of points increases above a certain value (Magenat, 2013). This filter is a part of the stack `ethzasl_icp_mapping`
- *Statistical Outlier Removal Filter:* This filter is used to remove the outliers which classified on the basis of distribution of point to neighbour distances in the dataset. This distribution of point to neighbour distances is assumed to be Gaussian. For every point in the dataset, the average distance from the neighbours is calculated. The number of points to be considered as neighbours can be varied. Now the points that have mean-distance inside the interval bounded by the global mean-distance and standard deviation are considered inliers. The points that lie outside the above interval are classified as outliers and are removed from the dataset. (PCL-Documentation, 2013b). This filter is available in PCL
- *Radius Outlier Removal Filter:* In this filter, the points which have a more than a specific number of neighbours are considered as inliers. For being considered a neighbour a point has to lie within a specified radius of the point for which neighbour search is being performed (PCL-Documentation, 2013a). The Figure 4.4 on the next page shows the concept of this filter. If the number of neighbours a point should have is specified to be 1, then the point a is filtered out. If the number of neighbour is specified to be 2, then points a and c are filtered out. The radius d can be specified according to the point-set. This filter is available in PCL

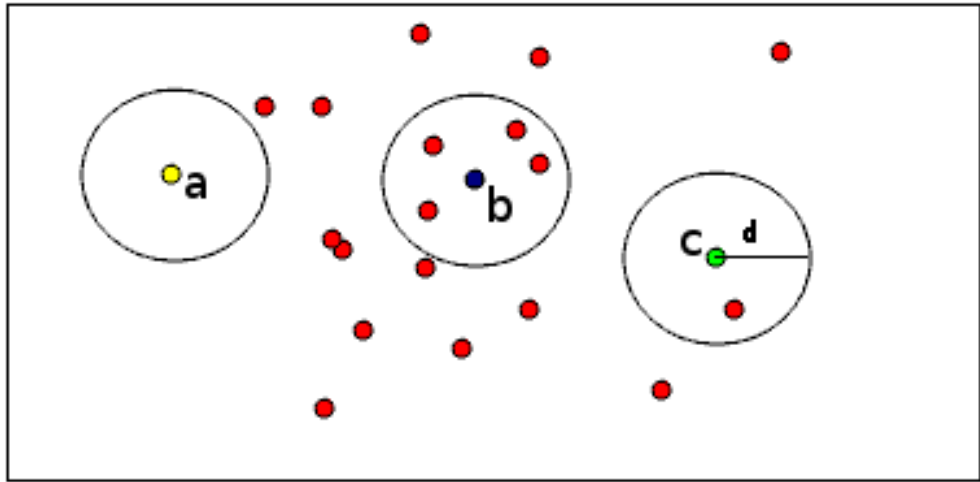


Figure 4.4: Radius Outlier Filtering. Figure adapted from PCL-Documentation (2013a)

- *XYZ Pass Through Range Filters:* After the maps were built, they were filtered in the X, Y and Z axis to make the data more presentable in the thesis. Data only within a certain range in each axis depending upon the scenario were kept. This filter is available in PCL

Chapter 5

Experiments and Results

5.1 Experimental set-up

In order to test the effectiveness of the algorithm, it was decided to first test it by trying to make a 3D map of a room which only had static objects followed by testing in unstructured indoor and outdoor cases with static objects. It was also decided to test the algorithm with and without odometry. Firstly the results of mapping without initial guesses are shown followed by the results of mapping which used odometry as an initial guess. The mapping with initial guesses was tested in four different scenarios namely: a) Indoor room, b) Indoor corridor, c) Indoor unstructured room and d) Unstructured outdoor scenario. The maps built in cases of corridor, indoor unstructured room and outdoor unstructured scenario were converted into octree maps format and bonsai tree map format and resulting compression results are shown. Bonsai tree map format is a compressed version of the octree map format provided by the `Octomap` framework.

5.2 Registration without an initial guess

The Figure 5.1 on the following page shows the registration result of half of the room. The mobile platform was moving in as straight line in the centre of the figure. A total of 9 scans were taken to get the result. The Figure 5.2 on the next page shows the registration result which failed to align all the scans.

This was the result of taking scans with larger distances between two of the consecutive scans resulting in less than required overlapping. It can be seen that ICP converged to a local minimum and the same object is seen twice in the registered point-cloud.

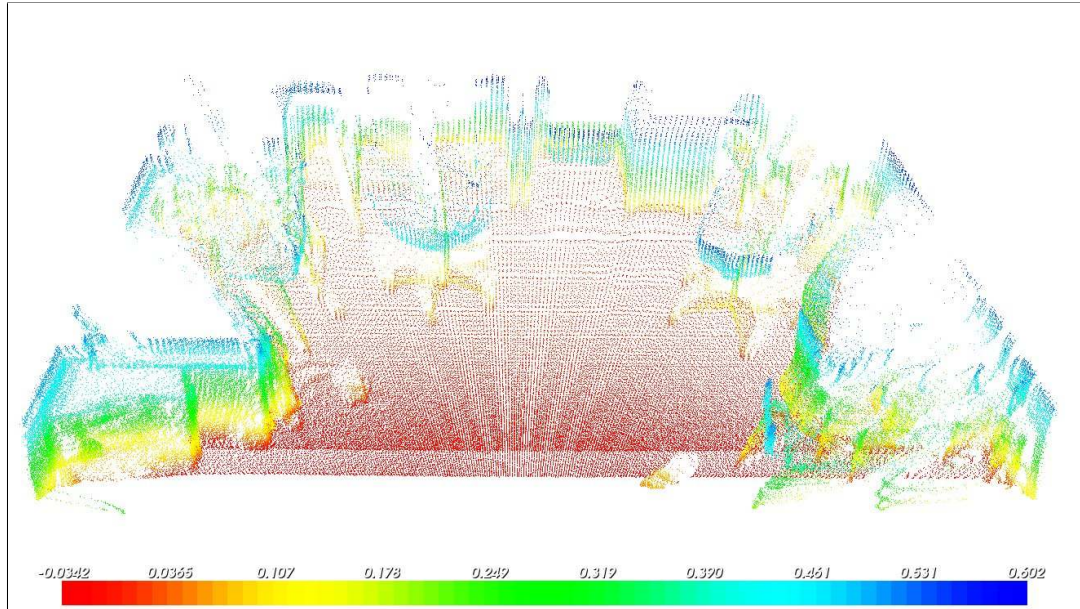


Figure 5.1: Registered aligned point-cloud of half of the room without odometry (Color legend unit in meters)

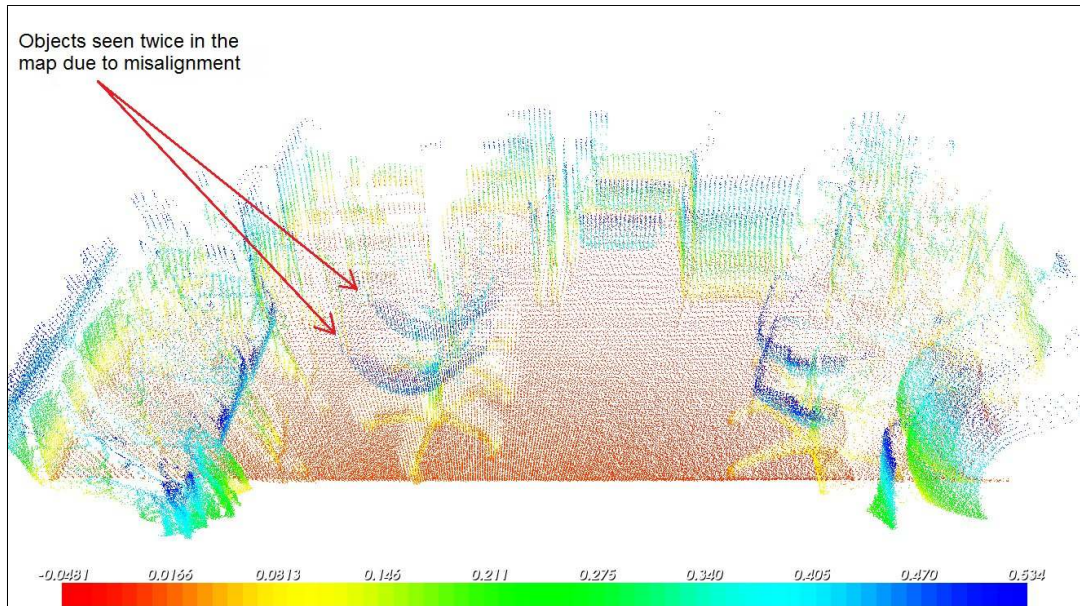


Figure 5.2: Registered unaligned point-cloud of half of the room without odometry (Color legend unit in meters)

The Figure 5.3 on the following page shows the photograph of the half of the



Figure 5.3: Photograph of half of the experimental room

room which was used for getting the 3D point-clouds shown in Figures 5.1 and 5.2 on the previous page.

5.3 Registration with initial guesses

This set of experiments was done by using odometry information as an initial guess for the transform minimizing the error function. The algorithm was used to map various scenarios both indoor and outdoor. They include:

1. Indoor room mapping
2. Indoor corridor mapping
3. Indoor unstructured room mapping
4. Outdoor mapping

5.3.1 Scenario 1: Indoor room mapping

In this scenario, two sets of experiments were performed. In the first one the indoor room was mapped by keeping the mobile platform at the same point,

while rotating it for approximately 360 degrees.

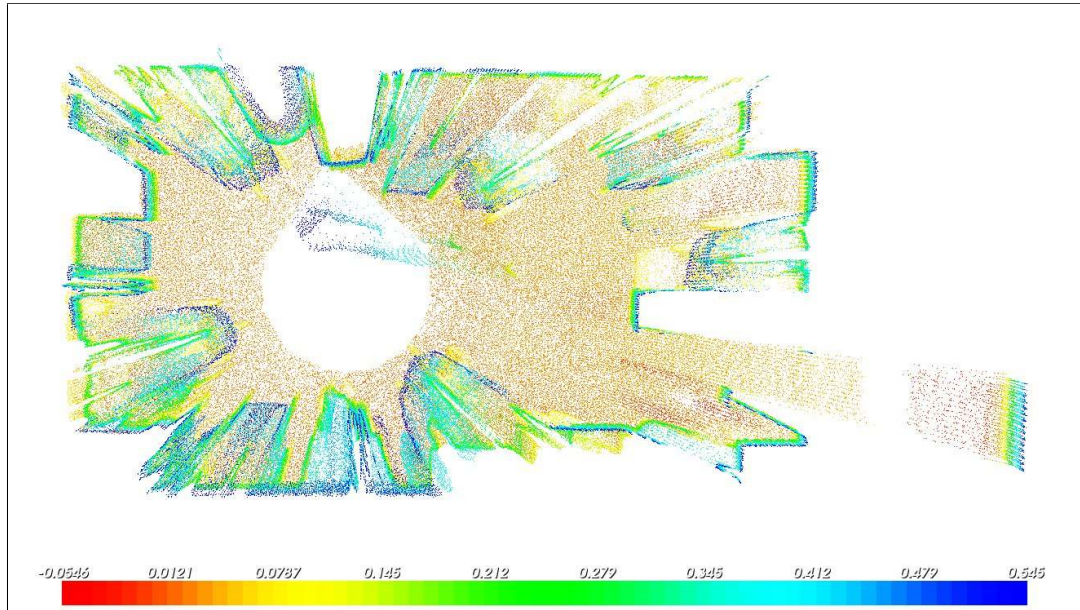


Figure 5.4: Registered point-cloud of the whole room. Top View (Color legend unit in meters)

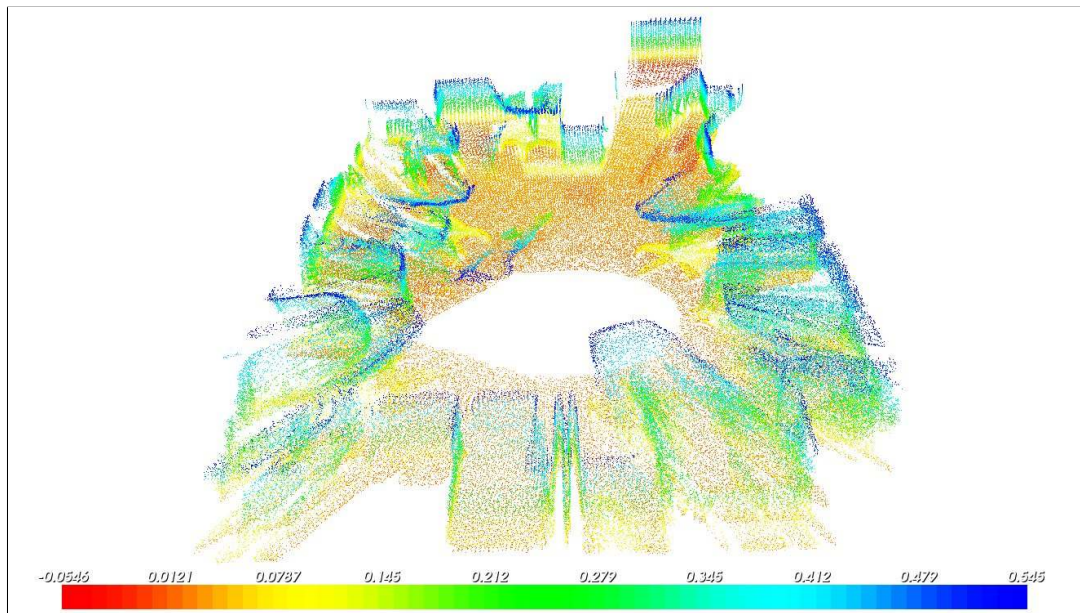


Figure 5.5: 3D point-cloud of the view relating to Figure 5.6 (Color legend unit in meters)

The Figure 5.4 shows the top-view 3D map of the room obtained. The Figure 5.6 on the next page shows the photograph of the room from one of corner of it and the Figure 5.5 is the same view in the 3D map.



Figure 5.6: Photograph of room from one of the Views

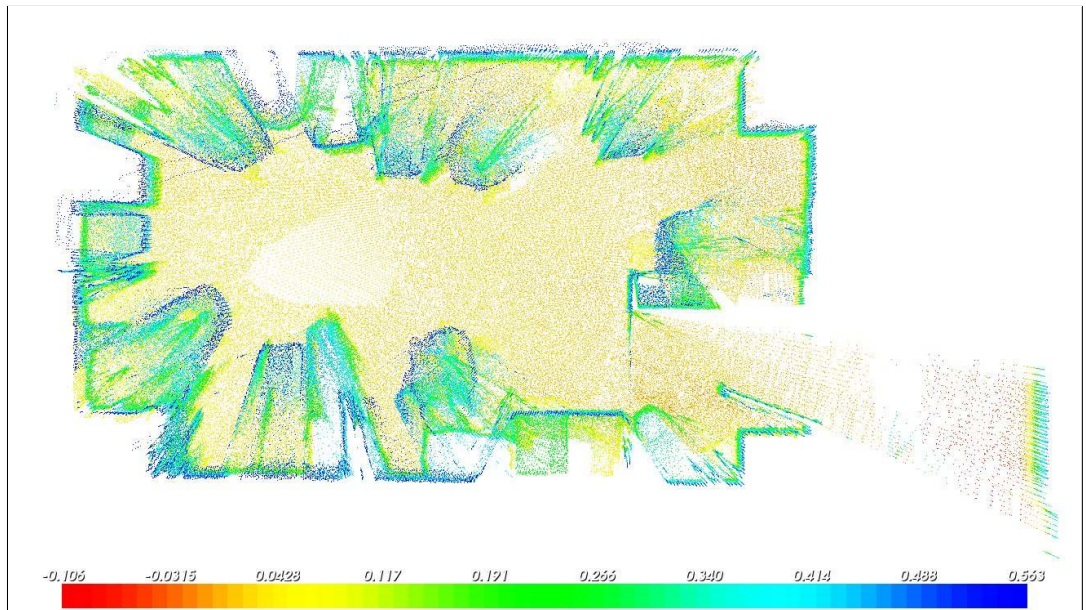


Figure 5.7: Top view of the map obtained by moving platform. (Color legend unit in meters)

In the second one, instead of rotating the mobile platform at the same place, it was moved around the room. The Figure 5.7 shows the top-view of the 3D map obtained by moving the mobile platform in the anticlockwise direction in the figure. The Figure 5.8 on the following page is the 3D map from the view of Figure 5.6.

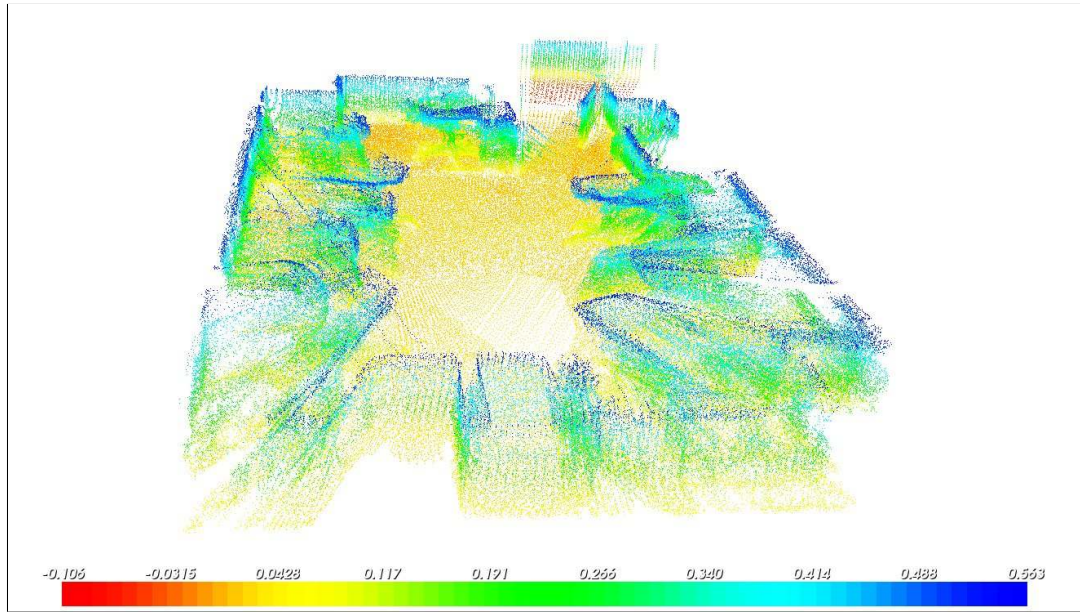


Figure 5.8: 3D map obtained by moving the platform. View related to Figure 5.6 (Color legend unit in meters)

5.3.2 Scenario 2: Indoor corridor mapping

For this scenario, the mobile platform was moved along a long corridor to make its 3D map. The Figure 5.10 on the next page shows the resulting map obtained. The Figure 5.11 on page 53 is the map in the octree representation. It was observed that there was misalignment in parts of the corridor where reflecting mirrors were present.

5.3.3 Scenario 3: Indoor unstructured room mapping

In order to make the experiment more challenging, the algorithm was tested in a room which was unstructured and has smooth surfaces. The results of the mapping experiments are shown in Figure 5.13 on page 54 and the corresponding octree map is show in Figure 5.14 on page 54.

5.3.4 Scenario 4: Outdoor mapping

On the surface of the moon, there would no features like walls to help in mapping. So the final experiment was done outdoors. The terrain similar to



Figure 5.9: Photograph of corridor scenario

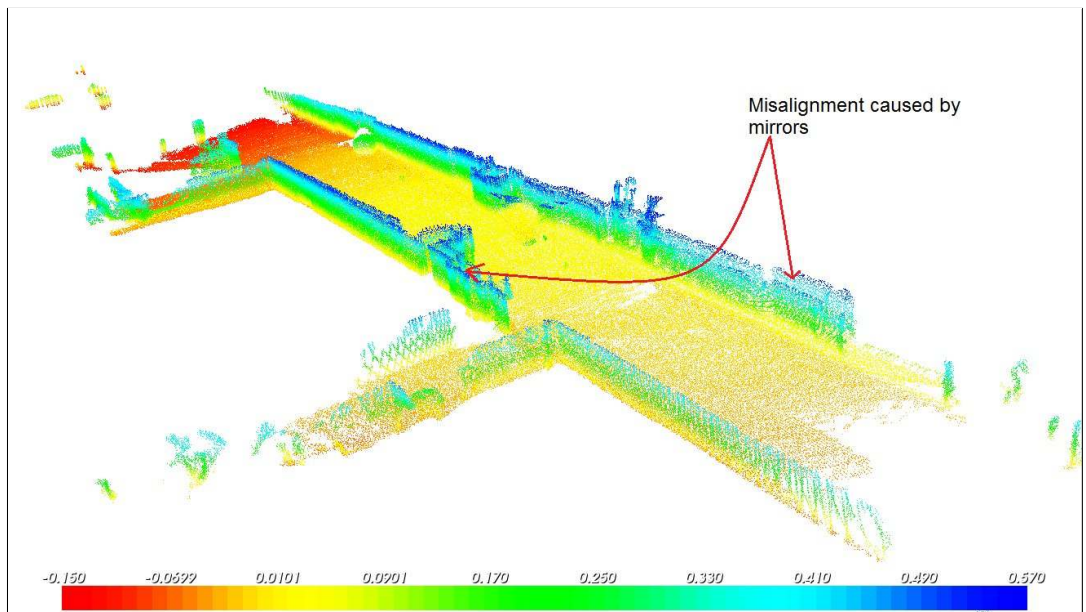


Figure 5.10: 3D map of the corridor (Color legend unit in meters)

moon was artificially created with the help of mats on various things like chairs. The resulting map obtained is shown in Figure 5.16 on page 55. The octree map of the outdoor scene is shown in Figure 5.17 on page 56.

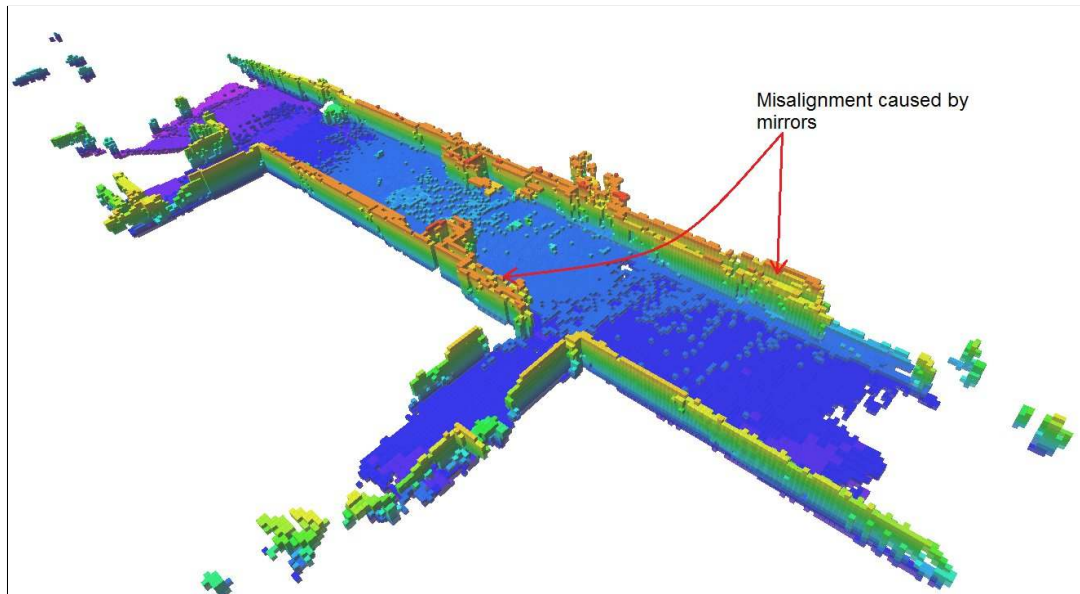


Figure 5.11: 3D octree map of the Corridor

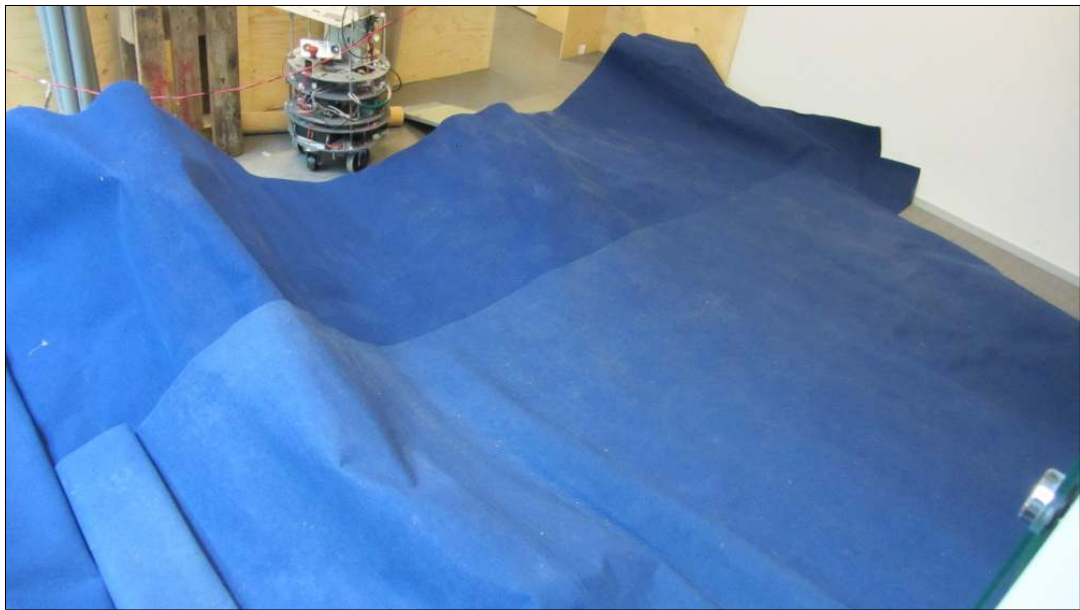


Figure 5.12: Photograph of unstructured room scenario

5.3.5 Octree compression results

One of the primary advantages of using octree map representation is the highly compact maps it enables to generate. The octree map is highly compact in spite of mapping free space. Another data format called *bonsai tree* structure is provided by the `Octomap` framework. The *bonsai tree* structure is even more compressed than a octree based map. It is similar to octree maps and is a compressed version of octree. The compression of the maps generated by the

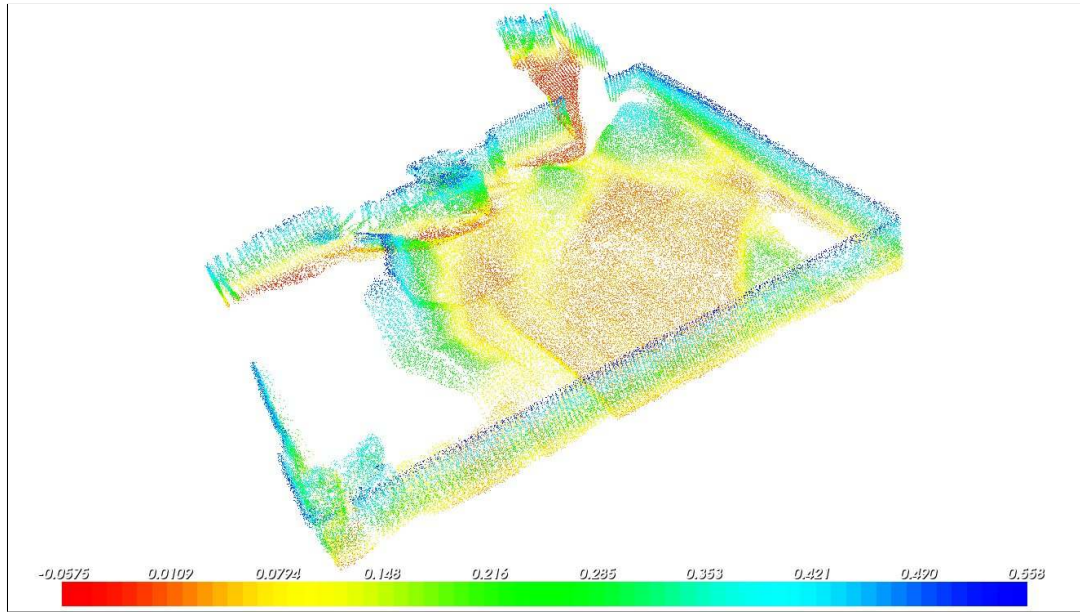


Figure 5.13: 3D Map of the unstructured room (Color legend unit in meters)

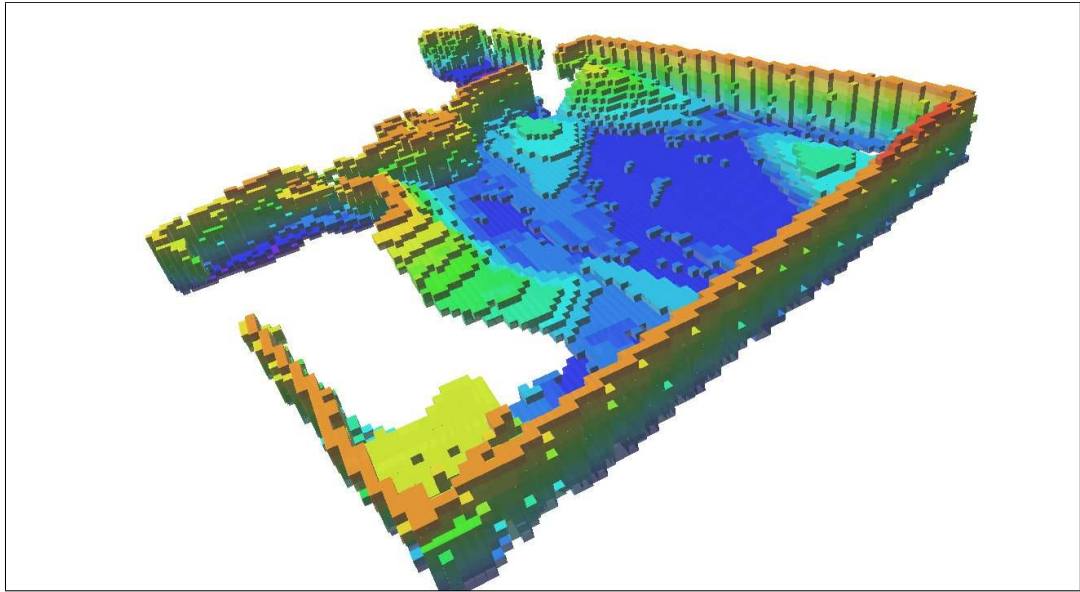


Figure 5.14: 3D Octree Map of unstructured room

Octomap framework is compared with the point-cloud data format (*.pcd) in the Table 5.1 on page 56. The compression for the three scenarios namely corridor, indoor unstructured room and outdoors by using the octree data format was 19 times, 23 times and 28 times respectively. The compression achieved by the *bonsai tree* data format was 239 times, 304 times and 362 times respectively for the three cases.



Figure 5.15: Photograph of outdoor scenario

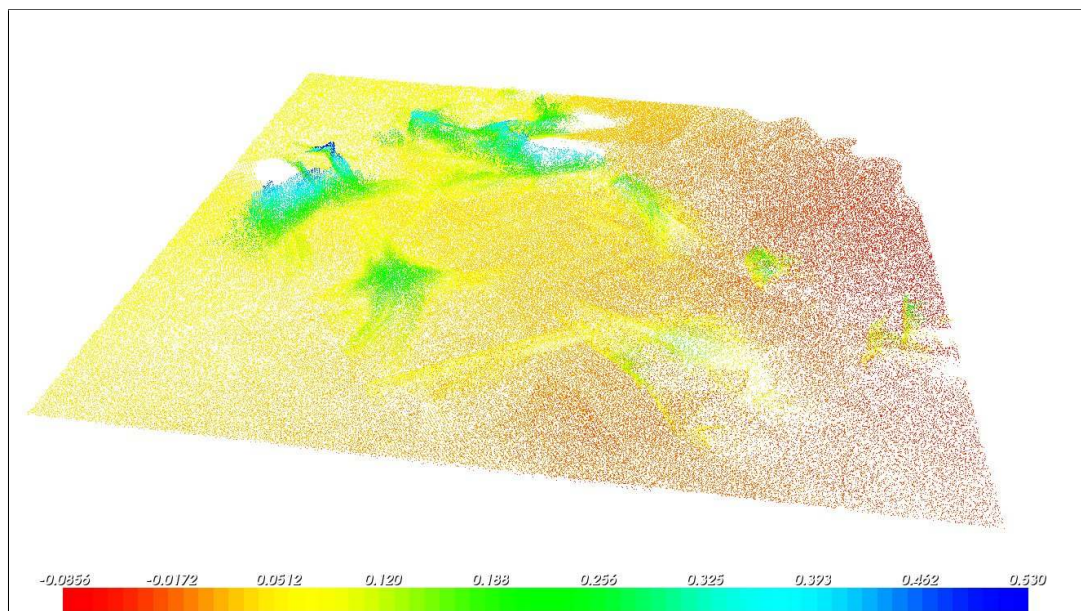


Figure 5.16: 3D Map of the outdoor scene

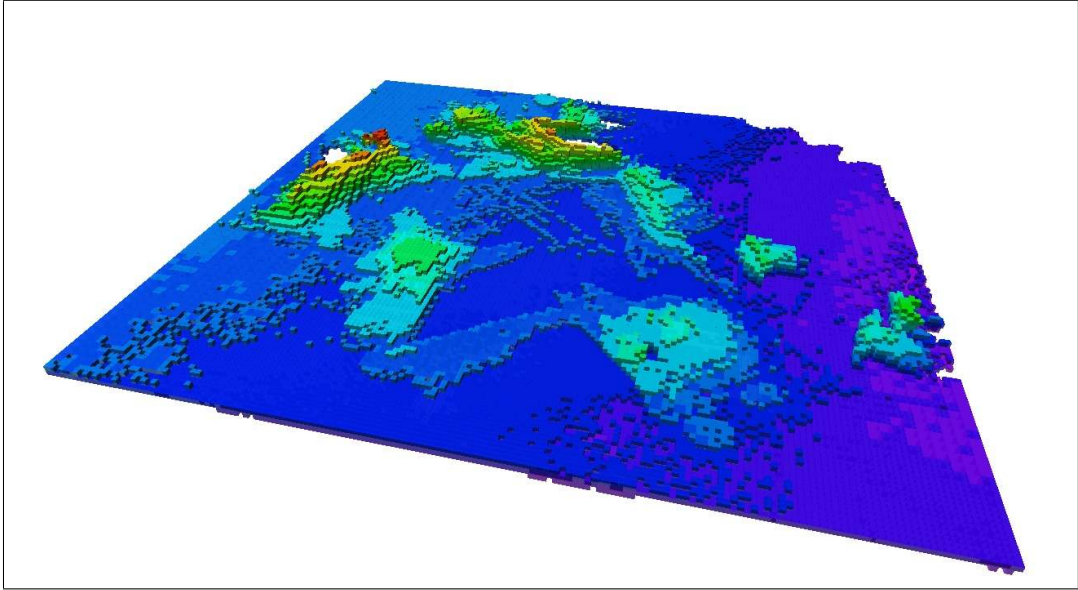


Figure 5.17: 3D Octree Map of outdoor scene

Table 5.1: Compression results of maps generated by Octomap framework

Map scenario	PCD file size	Octree file size	Bonsai Tree file size
Corridor	10.3 MB (10,319,693 bytes)	542.9 kB (542,930 bytes) (<i>Compression</i> = 19×)	43.3 kB (43,266 bytes) (<i>Compression</i> = 239×)
Indoor unstructured room	4.2 MB (4,205,717 bytes)	183.5 kB (183,529 bytes) (<i>Compression</i> = 23×)	13.8 kB (13,825 bytes) (<i>Compression</i> = 304×)
Outdoor	9.5 MB (9,532,047 bytes)	342.6 kB (342,619 bytes) (<i>Compression</i> = 28×)	26.4 kB (26,355 bytes) (<i>Compression</i> = 362×)

Chapter 6

Conclusions and Future Work

6.1 Conclusions

As we try to venture out more and more into the outer space away from earth, the need to have a human outpost on our nearest celestial neighbour, the Moon, becomes increasingly important. In order to increase our understanding of the Moon and to help in the initial ground work for the human outpost, it becomes absolutely essential that autonomous lunar rovers be developed. The broad aim of this thesis was to investigate the key aspect of creating maps that would help the lunar rover in navigating through the extreme environment of the Moon.

The primary aim of the thesis to develop a 3D mapping system for the prototype rover was successfully achieved. For reaching the objective stated above, a survey of the literature was first carried out to find out a suitable mapping approach from the various mapping approaches used in robotics. As commercially available 3D mapping systems are extremely costly, a low-cost 3D mapping system was developed using a 2D LIDAR scanner. A tilting mechanism consisting of a four-bar linkage controlled by a servo motor was built so that the LIDAR scanner could be tilted back and forth in the pitch axis. This tilting mechanism enabled the use of a 2D LIDAR scanner as a 3D LIDAR scanner, and thus obtain 3D terrain scans of the surroundings.

In order to build a globally consistent 3D map of the surrounding, all the terrain scans have to be combined into a common co-ordinate system. This process is called registration. One of the most popular algorithm for registration is the Iterative Closest Point (ICP) algorithm. The ICP algorithm was used to build a 3D map from the terrain scans. The ICP algorithm can be classified as a scan-matching approach to mapping. It was observed that the algorithm did not work satisfactorily when no initial guess was provided. The terrain scans did not align when the displacement between consecutive scans was large or when the two scans did not look almost the same. Then the algorithm was provided with an initial guess from the odometry information. The algorithm was tested in indoor scenarios and finally tested outdoors with obstacles similar to that of the moon. The algorithm worked satisfactorily in all the above test cases except one. This was probably due the mirrors in the corridor which had caused reflections and had thus resulted in misalignment in some parts of the map.

With a more detailed study of the ICP algorithm and by tweaking its parameters more to suit the application, its performance can be further increased. Hopefully the work done in this thesis laid the ground work for further work on 3D mapping for the prototype lunar rover. Some suggestions pointing in that direction are given in the next section.

6.2 Future Work

The primary objective of this thesis to develop a 3D terrain mapping and global map building system for a prototype lunar rover was successfully achieved. However, there are improvements in both the hardware and software that could be made to have an overall better system.

The scanning frequency of the 2D SICK LIDAR scanner was set to 25 Hz. The SICK LMS100 is capable of using scanning frequency up to 50 Hz. The scanning frequency should be increased to the highest value possible because this would enable the rover to obtain a faster terrain scan. The increase in scanning frequency would enable the rover to pitch the LIDAR scanner to faster

and hence obtain terrain scans faster. Presently there is no feedback from the servo to the overall system. This means that after the servo is commanded to a specific angle, the system has no way to know if the LIDAR scanner is pitched to the correct angle. One way to solve this problem is to use the encoder fit on the pitching axis and use position feedback from that encoder. The other alternative which would be better but costlier is to use the Dynamixel series of servos that have feedback circuitry built into them. This feedback circuitry allows for precise position, velocity and torque control of the servo.

The terrain scans for building a map were obtained in a stop-and-go fashion. This means that the rover had to stop to obtain a scan every time. This slows down the movement of the rover. If the LIDAR scanner is pitched fast relative to the forward movement of the rover then the rover could avoid stopping for getting scans. At present a single terrain takes approximately 10 seconds. The speed of the mobile platform while going straight was set to 0.2 m/s . With the parameters set for the ICP used in all experiments the mobile platform was able to move about 2.1 meters in the unexplored corridor scenario. This means the mobile platform would take approximately 20 seconds to travel a distance of around 2 meters i.e. 0.1 m/s approximately. Note that this is just a crude approximation for a particular scenario. The actual speed and distance that could be traversed between successive scans depends on various factors like curvature in the path chosen, minimum overlap set, convergence threshold, whether the region is explored or not and others. But it can be said that the time taken to get a terrain scans acts as one of the bottlenecks for the traversal speed.

The other thing that could be improved would be to reduce the number of points obtained from a terrain scan. The large number of points resulted in slower registration with ICP. The ICP library `libpointmatcher` has a varied set of data-filter that could be used not only for outlier rejection but greatly reduce the number of redundant points. These filters have to be tested for the required scenario and be set up accordingly in order to have the best performance.

Lastly, the system built for this thesis could be extended to deal with dynamic obstacle environment. The maps shown in Chapter 5 had only static obstacles. The Octomap framework used for building octree maps could be helpful for this purpose.

References

Adept-MobileRobots (2013). *ARIA*. [Online; accessed 24nd-July-2013].

URL: <http://robots.mobilerobots.com/wiki/ARIA> (p. 27)

ARDUINO (2013). *Arduino Mega 2560*. [Online; accessed 24th-April-2013].

URL: <http://arduino.cc/en/Main/arduinoBoardMega2560> (p. 21, 22)

ARRAS, K.O. (2003). *Feature-based robot navigation in known and unknown environments*. Ph.D. thesis. (p. 18)

BAJRACHARYA, M., MAIMONE, M.W., AND HELMICK, D. (2008). *Autonomy for Mars Rovers: Past, Present, and Future*. *Computer*, 41(12):44–50. (p. 2)

BARFOOT, T.D., DUPUIS, E., AND TONG, C.H. (2011). *3D SLAM for planetary worksite mapping*. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 631–638. (p. 4, 5)

BESL, P. AND MCKAY, H. (1992). *A method for registration of 3-D shapes*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256. (p. 36, 39)

BLAIR, S. (2011). *ROVERS RETURN*. *Engineering Technology*, 6(3):48–50. (p. 2)

CHEN, Y. AND MEDIONI, G. (1992). *Object modelling by registration of multiple range images*. In *Image and Vision Computing*, volume 10, pages 145–155. (p. 36, 39)

CHETVERIKOV, D., SVIRKO, D., STEPANOV, D., AND KRSEK, P. (2002). *The Trimmed Iterative Closest Point algorithm*. In *Pattern Recognition*,

2002. *Proceedings. 16th International Conference on*, volume 3, pages 545–548. (p. 43)

CRAWFORD, I., ANAND, M., COCKELL, C., FALCKE, H., GREEN, D., JAUMANN, R., AND WIECZOREK, M. (2012). *Back to the Moon: The scientific rationale for resuming lunar surface exploration. Planetary and Space Science*, 74(1):3–14. (p. 2)

DIGIUSEPPE, M., PIRICH, R., AND KRAUT, V. (2009). *Lunar regolith control and resource utilization. In 2009 IEEE Long Island Systems, Applications and Technology Conference*, pages 1–5. IEEE. (p. 3)

DUDEK, G. AND JENKIN, M. (2010). *Computational Principles of Mobile Robotics*. Cambridge University Press, New York, NY, USA, second edition edition. (p. 7, 8)

EVERETT, H.R. (1995). *Sensors for mobile robots: theory and application*. A. K. Peters, Ltd., Natick, MA, USA. (p. 7, 8)

FILLIAT, D. AND MEYER, J.A. (2003). *Map-based navigation in mobile robots: I. A review of localization strategies. Cognitive Systems Research*, 4(4):243–282. (p. ix, 11, 12, 13)

FOX, D., BURGARD, W., DELLAERT, F., AND THRUN, S. (1999). *Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In Proceedings of the 16th National Conference on Artificial Intelligence*. (p. 16)

FUKE, Y. AND KROTKOV, E. (1996). *Dead reckoning for a lunar rover on uneven terrain. In Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 411–416. IEEE. (p. 7)

GEMME, S., GINGRAS, D., SALERNO, A., DUPUIS, E., POMERLEAU, F., AND MICHAUD, F. (2012). *Pose Refinement Using ICP Applied to 3D LIDAR Data for Exploration Rovers. In International Conference on Artificial Intelligence, Robotics, and Automation in Space (iSAIRAS)*. (p. 19)

GRÜN, E., HORANYI, M., AND STERNOVSKY, Z. (2011). *The lunar dust environment. Planetary and Space Science*, 59(14):1672–1680. (p. 3)

- HORNUNG, A., WURM, K.M., BENNEWITZ, M., STACHNISS, C., AND BURGARD, W. (2013). *OctoMap: an efficient probabilistic 3D mapping framework based on octrees*. *Autonomous Robots*, 34(3):189–206. (p. viii, x, 40, 41, 42)
- HUNTRESS, W.T., MOROZ, V.I., AND SHEVALEV, I.L. (2003). *Lunar and Planetary Robotic Exploration Missions in the 20th Century*. *Space Science Reviews*, 107(3-4):541–649. (p. 2)
- ISHIGAMI, G., OTSUKI, M., AND KUBOTA, T. (2012). *Lidar-Based Terrain Mappnig and Navigation For A Planetary Exploration Rover*. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space*. (p. 4, 18)
- KYRKI, V. (2012). *Stereo Vision*. Lecture Slides, Machine Perception. (p. ix, 8)
- LACROIX, S., MALLET, A., BONNAFOUS, D., BAUZIL, G., FLEURY, S., HERRB, M., AND CHATILA, R. (2002). *Autonomous Rover Navigation on Unknown Terrains: Functions and Integration*. *The International Journal of Robotics Research*, 21(10-11):917–942. (p. 4, 18)
- LOW, K.L. (2004). *Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration*. Technical Report TR04-004, University of North Carolina at Chapel Hill. (p. 39)
- MAGNENAT, S. (2013). *ETHZ ASL ICP Chain Configuration*. [Online; accessed 22nd-July-2013].
URL: http://www.ros.org/wiki/ethzasl_icp_configuration
(p. 43, 44)
- MAIMONE, M., BIESIADECKI, J., TUNSTEL, E., CHENG, Y., AND LEGER, C. (2006). *SURFACE NAVIGATION AND MOBILITY INTELLIGENCE ON THE MARS EXPLORATION ROVERS*. In *Intelligent Space Robotics*, chapter SURFACE NA, pages 45–70. TSI Press. (p. 4, 5, 17, 18, 19)
- MASUDA, T., SAKAUE, K., AND YOKOYA, N. (1996). *Registration and integration of multiple range images for 3-D model construction*. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 1, pages 879–883. (p. 43)

MEYER, J.A. AND FILLIAT, D. (2003). *Map-based navigation in mobile robots: II. A review of map-learning and path-planning strategies. Cognitive Systems Research*, 4(4):283–317. (p. 14)

MILFORD, M. (2008). *Robot Navigation from Nature*. In *Robot Navigation from Nature*, volume 41 of *Springer Tracts in Advanced Robotics*, chapter Robotic Mapping Methods, pages 15–28. Springer Berlin Heidelberg, Berlin, Heidelberg. (p. 15, 16)

NASA (2013a). *Space Technology: Star Camera*. [Online; accessed 24nd-July-2013].

URL: http://nmp.nasa.gov/st6/TECHNOLOGY/star_camera.html (p. ix, 10, 11)

NASA (2013b). *Voyager: The Interstellar Mission*. [Online; accessed 15th-August-2013].

URL: <http://voyager.jpl.nasa.gov/> (p. 1)

NISHIDA, S.I. AND WAKABAYASHI, S. (2010). *A Plan for Lunar Exploration by using Robots*. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, pages 594–599. (p. 3)

NÜCHTER, A. (2009). *3D Robotic Mapping: The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom*. Springer Publishing Company, Incorporated, 1st edition. ISBN 3540898832, 9783540898832. (p. xii, 10, 36, 37, 38)

PARK, S.Y. AND SUBBARAO, M. (2003). *A fast point-to-tangent plane technique for multi-view registration*. In *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, pages 276–283. doi:10.1109/IM.2003.1240260. (p. x, 40)

PCL-DOCUMENTATION (2013a). *Removing outliers using a Conditional or RadiusOutlier removal*. [Online; accessed 22nd-July-2013].

URL: http://pointclouds.org/documentation/tutorials/remove_outliers.php (p. x, 44, 45)

PCL-DOCUMENTATION (2013b). *Removing outliers using a StatisticalOutlierRemoval filter*. [Online; accessed 22nd-July-2013].

- URL:** http://pointclouds.org/documentation/tutorials/statistical_outlier.php (p. 44)
- PCL-DOCUMENTATION (2013c). *What is PCL?* [Online; accessed 23nd-July-2013].
- URL:** <http://pointclouds.org/about/> (p. 23, 24)
- PIONEER (2013). *Pioneer 3-DX Datasheet*. (p. ix, 20)
- POMERLEAU, F. (2013). *Applied Registration for Robotics- Methodology and Tools for ICP-like Algorithms*. Ph.D. thesis, ETH, Zürich, Switzerland. (p. 38)
- POMERLEAU, F., COLAS, F., SIEGWART, R., AND MAGNENAT, S. (2013). *Comparing ICP variants on real-world data sets. Autonomous Robots*, 34(3):133–148. (p. 38, 39)
- POMERLEAU, F., MAGNENAT, S., COLAS, F., LIU, M., AND SIEGWART, R. (2011). *Tracking a depth camera: Parameter exploration for fast ICP. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3824–3829. (p. 38)
- QUIGLEY, M., CONLEY, K., GERKEY, B.P., FAUST, J., FOOTE, T., LEIBS, J., WHEELER, R., AND NG, A.Y. (2009). *ROS: an open-source Robot Operating System*. In *ICRA Workshop on Open Source Software*. (p. 22, 23)
- ROS-Wiki-Arduino (2013). *ROS: roserial arduino servo control tutorial*. [Online; accessed 05th-May-2013].
- URL:** http://www.ros.org/wiki/roserial_arduino/Tutorials/ServoController (p. x, 29)
- ROS-Wiki-laser-assembler (2013). *laser_assembler*. [Online; accessed 24nd-July-2013].
- URL:** http://www.ros.org/wiki/laser_assembler (p. ix, 27, 28)
- ROS-Wiki-laserfilters (2013). *laser_filters*. [Online; accessed 24nd-July-2013].
- URL:** http://www.ros.org/wiki/laser_filters (p. 27)
- Ros-Wiki-roserial (2013). *roserial*. [Online; accessed 24nd-July-2013].
- URL:** <http://www.ros.org/wiki/roserial> (p. 27)

ROS-Wiki-RST (2013). *robot_state_publisher*. [Online; accessed 24nd-July-2013].

URL: http://www.ros.org/wiki/robot_state_publisher (p. 27)

ROS-Wiki-tf (2013). *tf*. [Online; accessed 24nd-July-2013].

URL: <http://www.ros.org/wiki/tf> (p. 25)

ROS-Wiki-URDF (2013). *ROS: URDF and XML*. [Online; accessed 30th-May-2013].

URL: <http://www.ros.org/wiki/urdf/XML> (p. 32)

RUSU, R. AND COUSINS, S. (2011). *3D is here: Point Cloud Library (PCL)*. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. (p. 23)

SAARINEN, J. (2009). *A sensor-based personal navigation system and its application for incorporating humans into a human-robot team*. Ph.D. thesis, Helsinki University of Technology, Espoo, Finland. (p. 16, 17)

SEDGEWICK, R. AND WAYNE, K. (2013). *Lecture-3.3 Geometric Applications of BST*. Lecture Slides that accompany book Algorithms, 4th Edition[Online; accessed 11th-July-2013].

URL: <http://algs4.cs.princeton.edu/lectures/> (p. x, 39)

SEGAL, A., HAEHNEL, D., AND THRUN, S. (2009). *Generalized-ICP*. In *Proceedings of Robotics: Science and Systems*. Seattle, USA. (p. 38)

SOFTINTEGRATIONINC. (2013). *Interactive Four-Bar Linkage Position Analysis*. [Online; accessed 28th-May-2013].

URL: <http://www.softintegration.com/chhtml/toolkit/mechanism/fourbar/fourbarpos.html> (p. 31)

THRUN, S. (2002). *Robotic Mapping : A Survey*. In B. Lakemeyer, G. and Nebel, editor, *Exploring Artificial Intelligence in the New Millenium*, February. Morgan Kaufmann. (p. 11, 13, 14, 15, 16)

TRUCCO, E. AND VERRI, A. (1998). *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA. ISBN 0132611082. (p. 8)

VASAVADA, A.R., BANDFIELD, J.L., GREENHAGEN, B.T., HAYNE, P.O., SIEGLER, M.A., WILLIAMS, J.P., AND PAIGE, D.A. (2012). *Lunar equatorial surface temperatures and regolith properties from the Diviner Lunar Radiometer Experiment*. *Journal of Geophysical Research*, 117(June 2009):E00H18. (p. 3)

WULF, O. AND WAGNER, B. (2003). *FAST 3D SCANNING METHODS FOR LASER MEASUREMENT SYSTEMS*. In *International Conference on Control Systems and Computer Science (CSCS14)*, section 2, pages 3–8. (p. ix, 10, 21)

Appendix A

Four-bar linkage position analysis

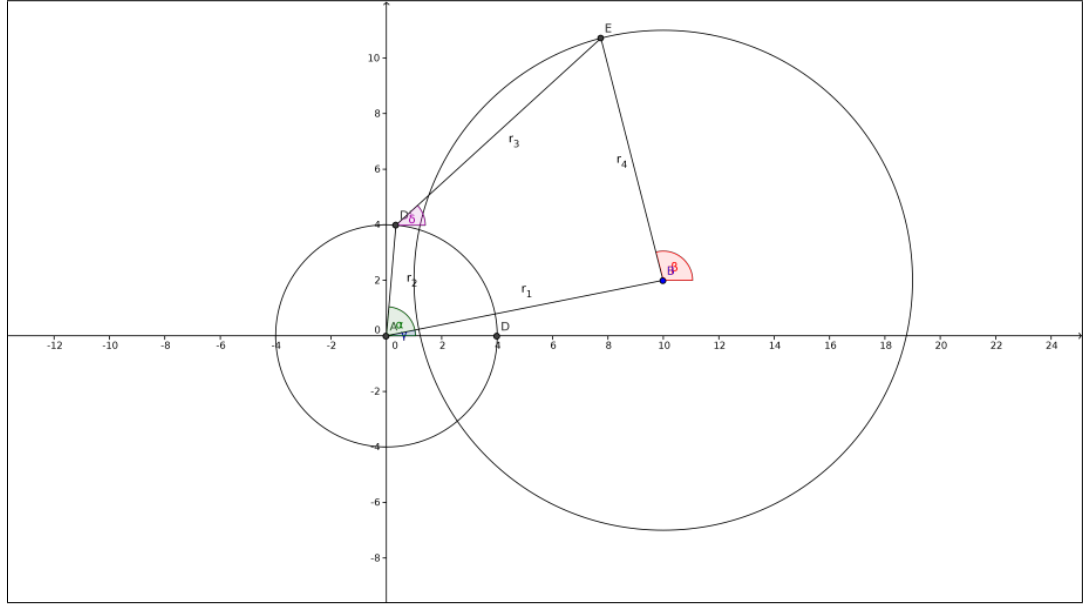


Figure A.1: A general four-bar mechanism

Let the lengths of the four links of the mechanism be r_1 , r_2 , r_3 , and r_4 as shown in the Figure A.1.

Then the position vectors of the four points A , B , E and D' by using complex number notation is $r_1 e^{i\gamma}$, $r_2 e^{i\alpha}$, $r_3 e^{i\delta}$ and $r_4 e^{i\beta}$ respectively, where $i = \sqrt{-1}$.

Now according to the Grashof condition for four-bar linkage mechanism,

$$r_2 e^{i\alpha} + r_3 e^{i\delta} = r_1 e^{i\gamma} + r_4 e^{i\beta} \quad (\text{A.1})$$

Rearranging the terms in the equation we get,

$$r_3 e^{i\delta} - r_4 e^{i\beta} = r_1 e^{i\gamma} - r_2 e^{i\alpha} \quad (\text{A.2})$$

Let

$$r_1 e^{i\gamma} - r_2 e^{i\alpha} = x + iy \quad (\text{A.3})$$

Also let $r_3 = R_3$ and $r_4 = -R_4$. Substituting these values of r_3 and r_4 and using A.3 we have,

$$x + iy = R_3 e^{i\delta} + R_4 e^{i\beta} \quad (\text{A.4})$$

By using euler's equation $e^{i\theta} = \cos(\theta) + i\sin(\theta)$ in Equation A.3 and Equation A.4 we get,

$$x + iy = r_1 (\cos \gamma + i \sin \gamma) - r_2 (\cos \alpha + i \sin \alpha) \quad (\text{A.5})$$

$$\therefore x + iy = r_1 \cos \gamma - r_2 \cos \alpha + i (r_1 \sin \gamma - r_2 \sin \alpha) \quad (\text{A.6})$$

$$\therefore x = r_1 \cos \gamma - r_2 \cos \alpha \quad (\text{A.7})$$

$$\therefore y = r_1 \sin \gamma - r_2 \sin \alpha \quad (\text{A.8})$$

We need β as a function of α . So α is independent variable. r_1, r_2, r_3, r_4 and γ are constants. Hence, the value of x and y can be calculated using Equations A.7 and A.8

$$x + iy = R_3 (\cos \delta + i \sin \delta) + R_4 (\cos \beta + i \sin \beta) \quad (\text{A.9})$$

$$\therefore x + iy = R_3 \cos \delta + R_4 \cos \beta + i (R_3 \sin \delta + R_4 \sin \beta) \quad (\text{A.10})$$

Equating the real and imaginary parts of Equation A.10 the value of $\cos \delta$ and $\sin \delta$ can be written in terms of x, y, R_3, R_4 and $\cos \beta$

$$x = R_3 \cos \delta + R_4 \cos \beta \quad (\text{A.11})$$

$$y = R_3 \sin \delta + R_4 \sin \beta \quad (\text{A.12})$$

$$\therefore \cos \delta = \frac{x - R_4 \cos \beta}{R_3} \quad (\text{A.13})$$

$$\therefore \sin \delta = \frac{y - R_4 \sin \beta}{R_3} \quad (\text{A.14})$$

Substituting the value of $\cos \delta$ and $\sin \delta$ in the trigonometric identity, $\sin^2 \delta + \cos^2 \delta = 1$ we get,

$$\left(\frac{x - R_4 \cos \beta}{R_3} \right)^2 + \left(\frac{y - R_4 \sin \beta}{R_3} \right)^2 = 1 \quad (\text{A.15})$$

$$\therefore (x - R_4 \cos \beta)^2 + (y - R_4 \sin \beta)^2 = R_3^2 \quad (\text{A.16})$$

$$\therefore x^2 + R_4^2 \cos^2 \beta - 2xR_4 \cos \beta + y^2 + R_4^2 \sin^2 \beta - 2yR_4 \sin \beta = R_3^2 \quad (\text{A.17})$$

$$\therefore x^2 + y^2 + R_4^2 (\cos^2 \beta + \sin^2 \beta) - R_3^2 = 2R_4 (x \cos \beta + y \sin \beta) \quad (\text{A.18})$$

$$\therefore x^2 + y^2 + R_4^2 - R_3^2 = 2R_4 (x \cos \beta + y \sin \beta) \quad (\text{A.19})$$

$$\therefore x \cos \beta + y \sin \beta = \left(\frac{x^2 + y^2 + R_4^2 - R_3^2}{2R_4} \right) \quad (\text{A.20})$$

Dividing both sides of Equation A.20 by $\sqrt{x^2 + y^2}$ we get,

$$\left(\frac{x}{\sqrt{x^2 + y^2}} \right) \cos \beta + \left(\frac{y}{\sqrt{x^2 + y^2}} \right) \sin \beta = \left(\frac{x^2 + y^2 + R_4^2 - R_3^2}{2R_4 \sqrt{x^2 + y^2}} \right) \quad (\text{A.21})$$

Now let us suppose that $\tan \phi = \left(\frac{y}{x} \right)$. Hence $\cos \phi = \left(\frac{x}{\sqrt{x^2 + y^2}} \right)$ and $\sin \phi = \left(\frac{y}{\sqrt{x^2 + y^2}} \right)$. Substituting these values in Equation A.21 we get,

$$\cos \phi \cos \beta + \sin \phi \sin \beta = \left(\frac{x^2 + y^2 + R_4^2 - R_3^2}{2R_4\sqrt{x^2 + y^2}} \right) \quad (\text{A.22})$$

$$\therefore \cos (\phi + \beta) = \left(\frac{x^2 + y^2 + R_4^2 - R_3^2}{2R_4\sqrt{x^2 + y^2}} \right) \quad (\text{A.23})$$

$$\therefore \phi + \beta = \cos^{-1} \left(\frac{x^2 + y^2 + R_4^2 - R_3^2}{2R_4\sqrt{x^2 + y^2}} \right) \quad (\text{A.24})$$

$$\therefore \beta = \phi + \cos^{-1} \left(\frac{x^2 + y^2 + R_4^2 - R_3^2}{2R_4\sqrt{x^2 + y^2}} \right) \quad (\text{A.25})$$

Since $\tan \phi = \left(\frac{y}{x} \right)$ we get the final expression for β as,

$$\boxed{\beta = \tan^{-1} \left(\frac{y}{x} \right) + \cos^{-1} \left(\frac{x^2 + y^2 + R_4^2 - R_3^2}{2R_4\sqrt{x^2 + y^2}} \right)} \quad (\text{A.26})$$

The Equation A.26, gives the value of the angle β , for the input angle α . R_3 and R_4 are constants while x and y can be found out using Equations A.7 and A.8 on page II respectively for a given value of α .

Appendix B

Flow diagrams

The flow diagram of the `arduino_node` and the `angle_publisher` is shown in Figure B.1 and Figure B.2 on page VI respectively.

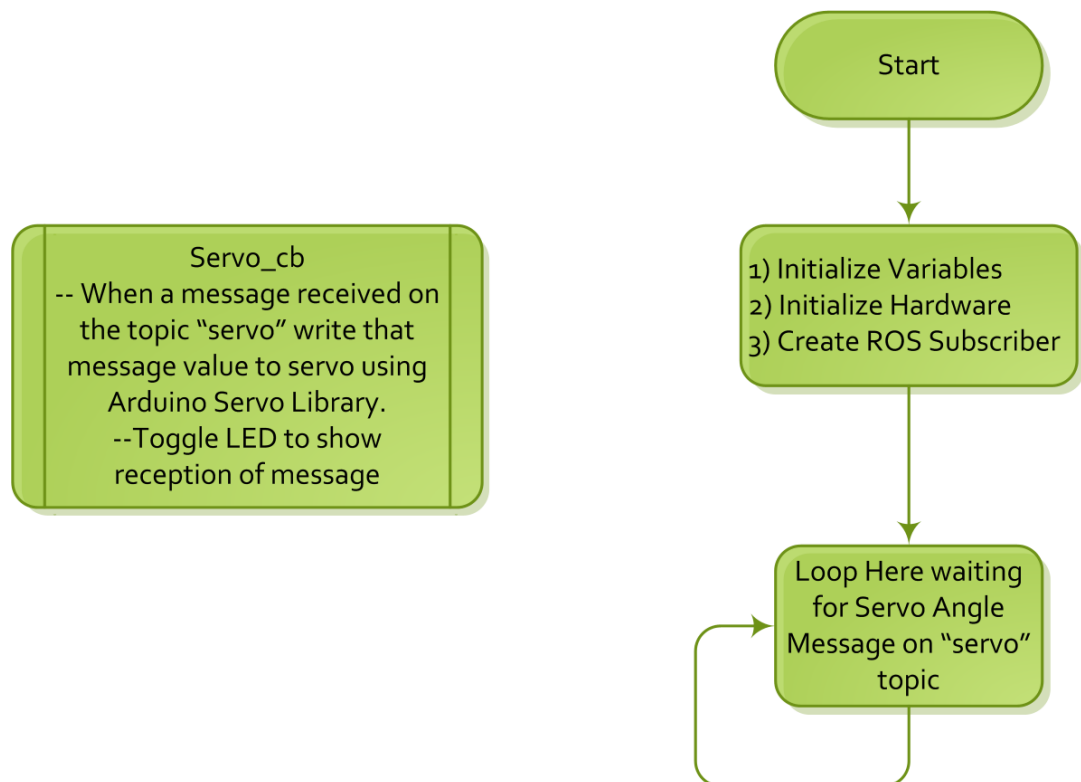


Figure B.1: Flow Chart of the Arduino Node

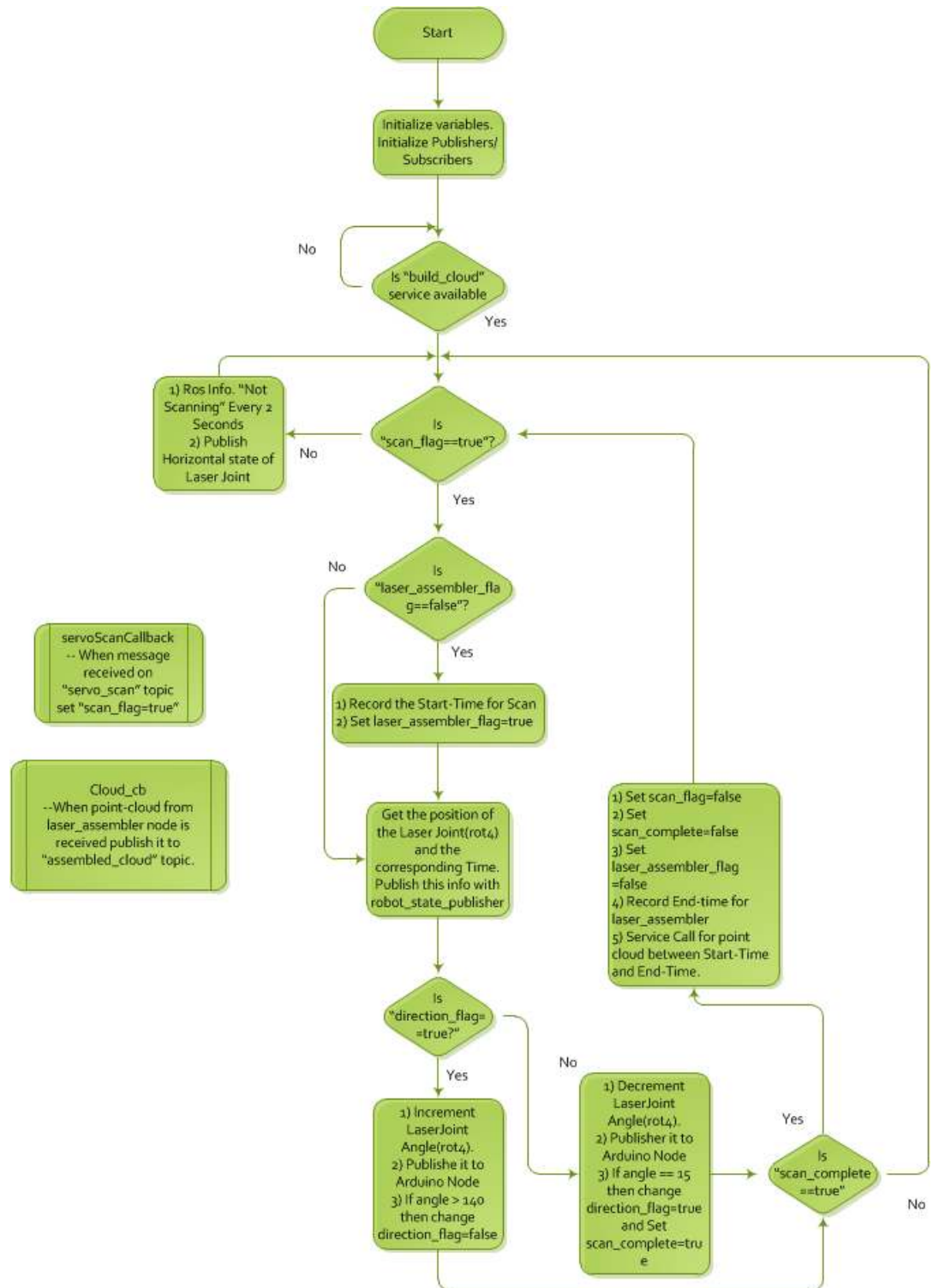


Figure B.2: Flow Chart of the Angle Publisher Node

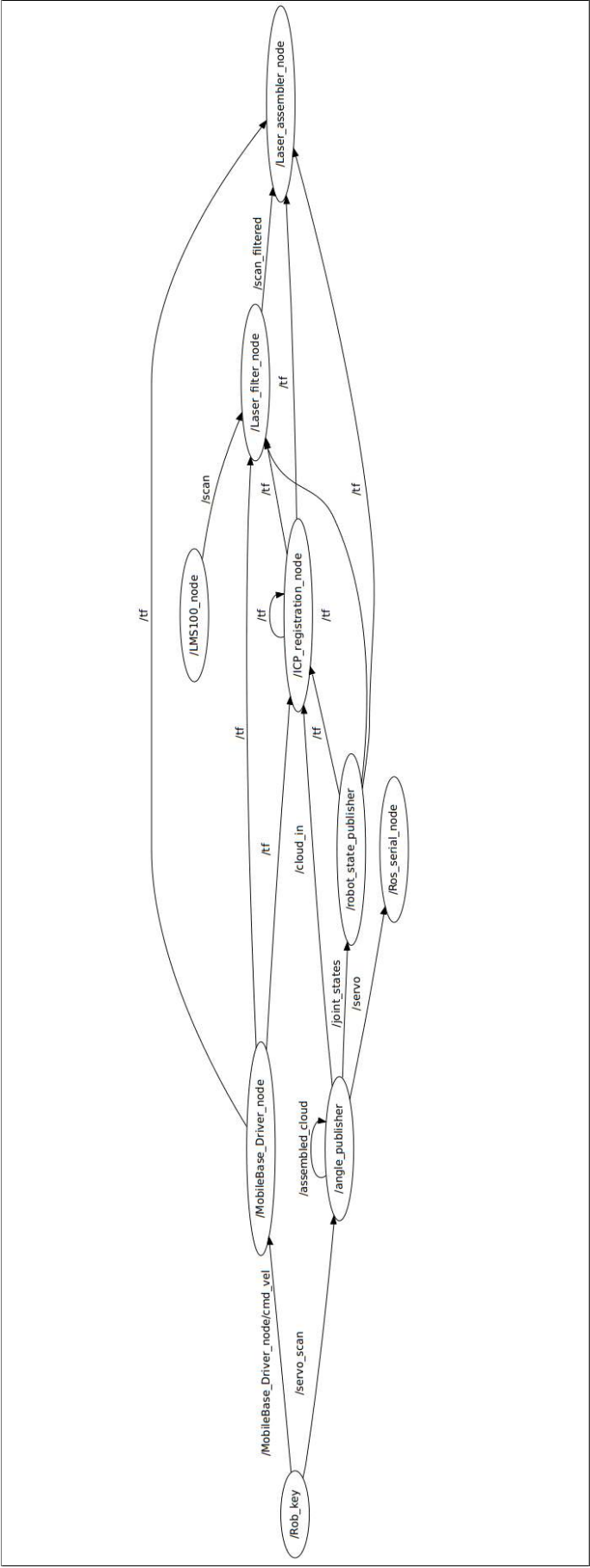


Figure B.3: Node Graph of the System